Optimización en sistemas multi-robot mediante Embodied Evolution

Autor: Pedro José Trueba Martínez

Tesis doctoral UDC / 2017

Directores:

Francisco Javier Bellas Bouza

Abraham Prieto García

Programa de doctorado en Computación[1]

[1]Programa regulado por el R.D. 1393/2007

Departamento de Computación

Tesis Doctoral

Optimización en sistemas multi-robot mediante Embodied Evolution

Pedro José Trueba Martínez

Directores:

Francisco Javier Bellas Bouza
Abraham Prieto García

2017

UNIVERSIDADE DA CORUÑA

D. Francisco Javier Bellas Bouza, Profesor Titular de Universidad del Departamento de Computación de la Universidade da Coruña,

D. Abraham Prieto García, Profesor Contratado Doctor del Departamento de Ingeniería Naval y Oceánica de la Universidade da Coruña,

CERTIFICAN:

Que la memoria titulada:

"Optimización en sistemas multi-robot mediante Embodied Evolution"

ha sido realizada por **D. Pedro José Trueba Martínez** bajo nuestra dirección en el Departamento de Computación, y constituye la Tesis que presenta para optar al grado de Doctor.

Fdo. Francisco Javier Bellas Bouza
Codirector de la Tesis Doctoral

Fdo. Abraham Prieto García
Codirector de la Tesis Doctoral

Resumen

En esta tesis se ha desarrollado una versión del algoritmo evolutivo Embodied Evolution (EE) que generaliza las existentes en el campo, con el objetivo de avanzar en la estandarización de este paradigma de forma que pueda ser estudiado de manera formal, y así conocer sus fortalezas y limitaciones. El algoritmo desarrollado en esta tesis se ha denominado "canónico"porque se ha diseñado tras el análisis detallado de los procesos básicos comunes a las diferentes variantes dentro de Embodied Evolution, de modo que únicamente contiene dichos procesos intrínsecos, y una parametrización de los mismos lo más general posible. El funcionamiento de este algoritmo canónico de Embodied Evolution se ha analizado en un conjunto de funciones teóricas representativas de los espacios de búsqueda en optimización colectiva, sobre los que se ha llevado a cabo un análisis de sensibilidad exhaustivo. Finalmente, las conclusiones del análisis teórico se han validado en una tarea real en la cual se ha podido comprobar la validez de la aproximación a la hora de optimizar la coordinación emergente del sistema multi-robot, tanto a nivel de rendimiento como a nivel de organización automática en especies, una propiedad fundamental de este paradigma.

Resumo

Nesta tese doutoral desenvolveuse unha version do algoritmo evolutivo Embodied Evolution (EE) que xeneraliza as existentes no campo, co obxectivo de avanzar na estandarización deste paradigma, de forma que poida ser estudado de maneira formal, e así coñecer as súas fortalezas e limitacións. O algoritmo desenvolvido nesta tese denominouse algoritmo canónico porque foi deseñado tras analizar detalladamente os procesos básicos comúns ás diferentes variantes dentro de Embodied Evolution, de modo que únicamente contén estes procesos intrínsecos, e una parametrización dos mesmos o máis xeral posible. O funcionamiento do algoritmo canónico de Embodied Evolution analizouse nun conxunto de funcións teóricas representativas dos espazos de búsqueda en optimización colectiva, sobre os que se realizou unha análise de sensibilidade exhaustiva. Finalmente, as conclusions da análise teorica validouse nunha tarefa real, na que se puido comprobar a validez da aproximación á hora de optimizar a coordinación emerxente do sistema multi-robot, tanto a nivel de rendemento como a nivel de organización automática en especies, unha propiedade fundamental neste paradigma.

Abstract

In this PhD thesis, a version of the Embodied Evolution (EE) algorithm has been developed that generalizes the existing version on the field, with the goal of advancing in the standardization of this paradigm such that it could be studied in a formal way and discover its strengths and limitations. The developed algorithm has been named "canonical" because it was designed after analyzing in detail the basic common processes of the different variants in Embodied Evolution, in a way that only contains the intrinsic processes and a parametrization as general as possible. The operation of this Embodied Evolution canonical algorithm has been analyzed in a set of theoretic functions that represent the different search landscapes in collective optimization, in which a sensibility analysis has been performed. Finally, the conclusions of the theoretic analysis has been validated in a real task in which the validity of the approximation has been confirmed through the successful optimization of the emergent coordination of the multi-robot system, both in performance and in automatic organization in species, a fundamental property in this paradigm.

Agradecimientos

En primer lugar, a los que más tengo que agradecer son a mis tutores Fran y Abraham que han hecho posible esta tesis y que me han ayudado desde el primer minuto. Sin duda les he hecho trabajar mucho pero creo que ha merecido la pena. Además de a mis tutores, también tengo que agradecer a otras personas sin las que este trabajo no hubiera sido posible: Richard, que propuso la idea de realizar un algoritmo canónico; Pilar, con la que hice el proyecto fin de carrera y aprendí de algoritmos evolutivos; y todos mis compañeros, tanto los de ahora como los anteriores. Siempre me han ofrecido su ayuda cuando la he necesitado.

En el aspecto personal, agradezco a mi familia que siempre ha estado de mi lado, tanto mi madre como mi hermana. Otra persona a la que estoy agradecido es a Ana, que es probablemente la persona con la que paso más tiempo y que me ha apoyado todo este tiempo. Gracias a todos.

Publicaciones

Durante el desarrollo de esta tesis han sido publicado los siguientes trabajos:

- P. Trueba, A. Prieto, P. Caamaño, F. Bellas, R.J. Duro (2011) Task-Driven Species in Evolutionary Robotic Teams. In: Ferrández J.M., Álvarez Sánchez J.R., de la Paz F., Toledo F.J. (eds) *Foundations on Natural and Artificial Computation. IWINAC 2011. Lecture Notes in Computer Science*, vol 6686. Springer, Berlin, Heidelberg

- P. Trueba, A. Prieto, F. Bellas, P. Caamaño, R.J. Duro (2012) Self-organization and Specialization in Multiagent Systems through Open-Ended Natural Evolution. In: Di Chio C. et al. (eds) *Applications of Evolutionary Computation. EvoApplications 2012. Lecture Notes in Computer Science*, vol 7248. Springer, Berlin, Heidelberg

- P. Trueba, A. Prieto, F. Bellas (2013) Distributed embodied evolution for collective tasks: parametric analysis of a canonical algorithm. In: *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, 37-38, 2013, ACM

- P. Trueba, A. Prieto, F. Bellas, P. Caamaño, R.J. Duro (2013) Specialization analysis of embodied evolution for robotic collective tasks, *Robotics and Autonomous Systems*, Volume 61, Issue 7, July 2013, Pages 682-693, ISSN 0921-8890, doi: 10.1016/j.robot.2012.08.005.

  (http://www.sciencedirect.com/science/article/pii/S0921889012001376)

- P. Trueba, A. Prieto, F. Bellas, R.J. Duro (2015) Embodied Evolution for Collective Indoor Surveillance and Location. In: Ferrández Vicente J., Álvarez-Sánchez J., de la Paz López F., Toledo-Moreo F., Adeli H. (eds)

Bioinspired Computation in Artificial Systems. IWINAC 2015. Lecture Notes in Computer Science, vol 9108. Springer, Cham

- P. Trueba, A. Prieto, F. Bellas, R. J. Duro (2015) Applying the canonical distributed Embodied Evolution algorithm in a collective indoor navigation task, *2015 International Joint Conference on Neural Networks (IJCNN)*, Killarney, 2015, pp. 1-8. doi: 10.1109/IJCNN.2015.7280807

- A. Prieto, F. Bellas, P. Trueba, R.J. Duro (2015) Towards the standardization of distributed Embodied Evolution, *Information Sciences*, Volume 312, 10 August 2015, Pages 55-77, ISSN 0020-0255,

  doi: 10.1016/j.ins.2015.03.044.

  (http://www.sciencedirect.com/science/article/pii/S0020025515002091)

- A. Prieto, F. Bellas, P. Trueba, R.J. Duro (2016) Real-time optimization of dynamic problems through distributed Embodied Evolution, *Integrated Computer-Aided Engineering*, vol. 23, no. 3, pp. 237-253, 2016

Índice general

Capítulo 1

Introducción y visión general

Esta tesis se centra en el desarrollo de un algoritmo evolutivo para la optimización de sistemas multi-robot. Por optimización, en este contexto, se entiende, por un lado, la obtención de la morfología del equipo de robots, tanto en número como en sus capacidades de sensorización y actuación, lo que se conoce como diseño morfológico del sistema. Por otro lado, también se debe optimizar el control de los robots con el objetivo de lograr un comportamiento coordinado que resuelva la tarea de manera colectiva. El uso de técnicas basadas en computación evolutiva para optimizar sistemas multi-robot no es nueva, ya que su gran capacidad de búsqueda en espacios de alta dimensionalidad los hace, a priori, muy adecuados para la resolución de problemas multi-componente. Pero la aproximación que aquí se plantea se diferencia en varios aspectos del tratamiento clásico de los algoritmos evolutivos en este contexto, todos ellos originados por el objetivo principal de esta tesis de optimizar sistemas multi-robot que puedan ser transferidos a la realidad de manera directa. Primeramente, se busca que el algoritmo no limite el tipo de morfología del sistema, de tal modo que no se reduzca al uso de robots homogéneos o con un número predeterminado de elementos. Además, se pretende que la solución al problema sea totalmente descentralizada, es decir, que no dependa de la existencia de canales de información centrales, sino que los robots deben ser totalmente autónomos en su operación. Finalmente, el algoritmo que se propone, parte de que la evolución de los individuos se

realiza en tiempo de ejecución mientras éstos interactúan en su entorno, de modo que el sistema sea tolerante a cambios. Con estas premisas de partida, en el Grupo Integrado de Ingeniería de la UDC se ha venido desarrollando desde hace años un algoritmo denominado ASiCo (Asynchronous Situated COevolution), basado en el paradigma evolutivo "Embodied Evolution" (Embodied Evolution (EE)), que se fundamenta también en estos requisitos de diseño. El algoritmo ASiCo ha mostrado su gran potencial en diversas aplicaciones prácticas de optimización en espacios de búsqueda colectivos, pero también ha quedado patente una alta sensibilidad a su parametrización y a las características de cada problema particular. En este sentido, las diferentes variantes del paradigma EE muestran una problemática similar, y por esta razón, en esta tesis se ha llevado a cabo el desarrollo de una versión "canónica" de un algoritmo de EE, es decir, una versión que identifique los procesos básicos del paradigma, los aisle del entorno de aplicación del sistema, y los parametrice de la manera más sencilla posible. Una vez desarrollado este algoritmo canónico de EE, se ha llevado a cabo un análisis y caracterización formal como etapa indispensable a la hora de ser aplicado con rigor en este dominio.

Para que un conjunto de robots realice un comportamiento colectivo que resuelva una tarea no trivial es necesaria una coordinación entre ellos, ya que simplemente la interacción libre de los mismos no es suficiente para que surja este tipo de comportamiento organizado. Los robots pueden compartir información y aprovechar las capacidades de otros robots, haciendo que el comportamiento colectivo sea más que la simple agregación de los individuales. La obtención de coordinación en sistemas multi-robot es compleja, ya que se debe diseñar el controlador individual de cada robot buscando una respuesta a nivel colectivo, teniendo en cuenta que, en general, no existe una relación directa entre comportamientos individuales y comportamiento colectivo. Asimismo, el control individual depende de la morfología del robot, lo cual también condiciona el comportamiento colectivo. Por este motivo, la optimización de la coordinación se ha tratado habitualmente como un problema separado de la optimización de la morfología, ya que así se reduce la complejidad del espacio de búsqueda. Otra discusión sería si esta es la forma ideal de afrontar el problema, ya que un diseño morfológico óptimo del sistema multi-robot, podría simplificar el control. En el marco de esta tesis doctoral, se ha impuesto como requisito de diseño para el algoritmo de optimización que posibilite también la obtención de la morfología del sistema, pero no se ha estudiado este aspecto. Por tanto, se ha centrado el esfuerzo en desarrollar un algoritmo "canónico"de EE que optimice el control

coordinado a partir de una morfología fija.

Podemos distinguir dos grandes grupos de aproximaciones a la hora de coordinar sistemas multi-robot: coordinación intencional (también denominada explícita) o coordinación emergente (también denominada implícita). En la coordinación intencional, los comportamientos individuales están normalmente predefinidos, y el objetivo se centra más en cómo asignar tareas al grupo, utilizando para ello comunicaciones explícitas entre robots. En la coordinación emergente no se predefinen ni los comportamientos ni la forma en que éstos pueden organizarse. La coordinación intencional tiene el problema de que no es escalable y es abordable únicamente para un grupo reducido de robots, mientras que con coordinación emergente se aprovechan las características de flexibilidad, robustez y escalabilidad de los sistemas colectivos. Por contra, la coordinación intencional puede quedar fácilmente atrapada en soluciones triviales al tener que solucionar problemas muy complejos que pueden contener numerosas ambigüedades. Esta tesis ha seguido una aproximación con coordinación emergente ya que no está sujeta a las restricciones de la coordinación intencional a la hora de su implantación práctica, por lo que proporciona un campo de desarrollo más abierto.

En la coordinación emergente, la mayoría de las aproximaciones parten de una inspiración biológica. Por un lado, debemos destacar la robótica de enjambre, que está basada en el comportamiento de sistemas colectivos naturales como, por ejemplo, los insectos, los bancos de peces, las colonias de abejas o las bandadas de pájaros. En robótica de enjambre los sistemas multi-robot están formados por robots muy simples pero en gran número, y la coordinación surge de la interacción entre estos elementos simples. Por otro lado, podemos encontrar métodos basados en aprendizaje por refuerzo, en los que se introducen penalizaciones y recompensas a cada robot después de cada secuencia de acciones que realizan. De esta forma modifica gradualmente su comportamiento hacia el comportamiento esperado. Estos algoritmos se han usado para obtener comportamientos en sistemas de un solo robot pero también en sistemas multi-robot, abordando las dificultades que surgen al realizar este paso: un incremento del espacio de estados y el problema de la asignación individual de la recompensa.

Finalmente, debemos hablar de los algoritmos evolutivos, que es el campo donde se enmarca esta tesis. Existen diferentes formas de aplicar evolución en sistemas multi-robot, pero se puede diferenciar, en un primer término, según

la forma de representación y de codificación de los controladores en los genotipos. Las primeras aproximaciones, y las más sencillas, se basan en un equipo homogéneo: se crea una población con n individuos donde cada uno codifica los parámetros de un controlador, y para evaluarlo, se clona en cada uno de los robots del equipo. Para evolucionar este controlador, se requiere definir una función de calidad que mediría el rendimiento del equipo en el problema a resolver, y en base a este criterio, se producirían sucesivamente mejores individuos hasta un obtener en equipo de controladores óptimo. La definición de la función de calidad es una de las claves de que el proceso funcione, y el cómo diseñar funciones de calidad adecuadas es uno de los problemas abiertos en robótica evolutiva multi-robot. Esta aproximación tradicional de obtención de los controladores homogéneos es un proceso de evolución off-line, ya que hasta que no finalice la evolución, el sistema multi-robot no empieza su operación en el entorno. Antes de esto necesita una fase de evolución en la que cíclicamente se forman equipos y se sustituyen todos los controladores. Solo cuando se hayan obtenido unos controladores óptimos, el sistema comienza su operación.

Cuando el problema, en vez de formularse con un equipo homogéneo, se formula con un equipo heterogéneo en el que cada uno de los controladores puede ser diferente, la forma más básica de aplicar un algoritmo evolutivo es codificar todo el equipo en un mismo genotipo del evolutivo. Esta aproximación tiene la desventaja de que la complejidad escala exponencialmente al aumentar en número de robots, con lo cual solo es adecuada con equipos poco numerosos. Otra aproximación consiste en evolucionar los controladores en poblaciones diferentes del mismo algoritmo evolutivo. Esta última estrategia hace surgir el concepto de co-evolución. La co-evolución es un tipo de evolución en donde la calidad de los individuos depende de su interacción con otros. Según la naturaleza de esta interacción, se puede diferenciar entre co-evolución competitiva y cooperativa. En co-evolución competitiva, los individuos ganan calidad frente a otros individuos que la pierden. Por contra, en co-evolución cooperativa los individuos obtienen recompensas o penalizaciones actuando conjuntamente como equipo. Desde el punto de vista de los dominios multi-robot, la co-evolución es muy interesante porque es una herramienta adecuada para abordar la explosión del espacio de búsqueda a la hora de optimizar los parámetros para un equipo de robots. El ejemplo más ilustrativo de co-evolución, y de los más estudiados, es el problema del depredador-presa.

Algunos ejemplos de co-evolución cooperativa se han desarrollado en tareas

de construcción colectiva, exploración colectiva, o búsqueda de puntos de interés. La co-evolución competitiva también se ha estudiado desde el punto de vista de la teoría de juegos, como por ejemplo, el dilema del prisionero. Una ventaja de la co-evolución es que se produce una dinámica que no ocurre en evolución clásica, que se conoce como carrera de armas evolutiva, en las que los individuos al co-evolucionar juntos, se adaptan mejorando relativamente uno frente al otro sucesivamente, pero produciendo una mejora en términos absolutos con una complejidad que a veces la evolución clásica no es capaz de obtener.

Existe toda una familia de algoritmos que siguen los principios de la co-evolución cooperativa que se denominan Cooperative Coevolutionary Algorithms (CCEAs), y se han aplicado tanto a problemas de optimización general con alta dimensionalidad como a sistemas multi-robot, para los que son adecuados debido a su estructura multi-componente. Estos algoritmos siguen una filosofía divide y vencerás ya que la evolución se realiza en varias poblaciones aisladas en paralelo, descomponiendo el problema en problemas más simples. Para sistemas multi-robot, se evoluciona el controlador de cada uno de los robots en poblaciones diferentes y aisladas. El número de poblaciones puede ser igual al número de robots o menor que el número de robots, en cuyo caso, varios robots se evolucionan en la misma población. A la hora de evaluar un controlador, se deben formar equipos con el resto de controladores de otras poblaciones para evaluarse conjuntamente. Las desventajas de esta técnica son varias: la complejidad se dispara al aumentar el número de robots en el sistema (a cada robot le corresponde una población), es difícil asignar la calidad a un individuo al ser evaluado en diferentes combinaciones de equipo y se debe establecer a priori el número de poblaciones, para lo que se debe conocer la descomposición del problema a resolver.

La complejidad se dispara porque el número de equipos posibles (combinaciones de individuos de diferentes poblaciones) a evaluar aumenta exponencialmente con el número de robots y realizar todas esas evaluaciones es costoso en tiempo computacional. Una consecuencia de este nivel de complejidad es que estos algoritmos no permiten una evolución en tiempo real y tienen que operar en simulaciones off-line, ya que la forma de evaluar a los individuos es formando equipos de manera síncrona, cambiando todo el equipo de controladores simultáneamente, para evaluar a todos los individuos, lo que implica reiniciar el escenario. La asignación de calidad es un problema en los CCEAS porque las evaluaciones dependen del contexto, y un individuo con buena calidad puede

ser evaluado con un equipo de mala calidad que no le permitiría obtener una valoración justa. Existen algunos trabajos que tratan este problema de cómo asignar calidad individual, por ejemplo, intentando medir la contribución real del robot al equipo o evaluando un individuo con un equipo y a su equipo sin él. Respecto al problema de conocer a priori el número de poblaciones, si se conoce la descomposición del problema, estos algoritmos son adecuados ya que cada población puede especializarse en cada sub-problema optimizando así la solución colectiva. De ahí que la especialización sea un aspecto a estudiar en estas aproximaciones y que también tenga relevancia en esta tesis. Como a veces no es posible conocer a priori dicha descomposición, existen algoritmos de este tipo en los que el número de poblaciones se modifica a lo largo de la evolución, creando o destruyendo poblaciones según corresponda con el estado actual de la evolución pero estas aproximaciones son muy complejas y no son aplicables a problemas reales.

Una técnica que busca dar solución a los problemas anteriores de los CCEAS es la denominada "Embodied Evolution" (EE) que propusieron Watson y Ficici en 1999, como una aproximación más aplicable a problemas multi-robot reales, ya que se basa en que la población que se evoluciona esta formada directamente por los robots reales que "habitan.[el] problema. Con ello, resuelve de manera innata dos problemas básicos de las técnicas evolutivas comunes: la transferencia de la simulación al mundo real y el elevado tiempo de cómputo de las evaluaciones. En el primer caso porque la evolución se lleva a cabo con los robots reales realizando la tarea mientras evolucionan (es un proceso "on-line", adaptándose en tiempo real) y en el segundo porque cada robot lleva incrustado un individuo que es el que se está evaluando en lugar de una población entera.

A pesar de que existen trabajos previos en robótica evolutiva en los que el evolutivo se encuentra incrustado en el cuerpo del robot, el algoritmo de Watson fue la primera aproximación distribuida de "embodied evolution" aplicado a sistemas multi-robot. En Embodied Evolution, los robots interactúan e intercambian información genética de forma asíncrona y local, ya que esta transmisión está basada en encuentros físicos en el entorno. Esto es una diferencia significativa respecto a un algoritmo evolutivo clásico donde la reproducción de individuos la realiza el algoritmo de forma centralizada, los procesos son síncronos y se llevan a cabo las etapas de la evolución secuencialmente: evaluación, reproducción y reemplazo.

En un algoritmo de EE, la presión selectiva, es decir, que los individuos con

mejor calidad propaguen con mayor frecuencia su código genético, se introduce a través del concepto de energía. Un robot consume energía pero también obtiene energía como una recompensa (definida por el diseñador) cuando se realiza de forma satisfactoria la tarea que se quiere resolver. Los robots que pierden toda su energía son reemplazados, mientras que los robots que tienen más energía pueden permanecer en el entorno más tiempo para transmitir su código genético a otros individuos. En la implementación original del algoritmo EE, la transmisión de material genético se produce siguiendo el algoritmo PGTA, que transmite genes de un robot a otro de forma probabilística, con una probabilidad proporcional a la energía del individuo que envía. Otra característica del algoritmo de EE es que es "open-ended", es decir, el sistema no tiene un límite de tiempo para evolucionar, sino que los hace indefinidamente adaptándose en tiempo real al entorno hasta que se detenga de manera explícita.

La línea de EE comenzada por el algoritmo de Watson et al. ha experimentado un gran auge en la última década, con diversas variaciones del algoritmo original que han sido aplicadas a diferentes tipos de problemas colectivos, aunque fundamentalmente en sistemas multi-robot, tanto reales como simulados. Existen dos grandes tendencias dentro de EE, la aproximación de EE distribuida (dEE) similar a la del algoritmo original, o la versión encapsulada (eEE), en la que los robots en vez de llevar dentro de su cuerpo un solo genotipo llevan una población entera. La versión encapsulada funciona como una serie de islas evolucionando en paralelo y en tiempo real, en las que se evalúan los individuos conjuntamente de forma asíncrona. La ventaja es que cada robot tiene variedad genética sin tener que comunicarse con ningún otro robot, pero con el inconveniente de que se deben utilizar mecanismos de reparto de tiempo para evaluar a todos los individuos de la población que almacena cada robot. Por otra parte, la versión distribuida es mucho más escalable y se beneficia a la hora de aumentar el número de robots, ya que de esta forma se garantiza una variedad genética distribuida sobre toda la población.

Dentro de la línea de dEE, en el Grupo Integrado de Ingeniería de la UDC se ha venido desarrollando el algoritmo ASiCO (Asynchronous Situated Coevolution). Al igual que el algoritmo de Watson, es un algoritmo de EE distribuido en el que cada robot lleva consigo un embrión que pasa a tomar el control del robot cuando el controlador actual del robot se reemplaza. Al contrario que en el algoritmo de Watson en el que se transmitían porciones del genotipo a otros individuos, en este algoritmo se transmite información genética para formar

embriones, que son genotipos que quedan almacenados para que individuos potenciales tomen control del robot cuando ocurra el reemplazo. Esta transmisión genética se produce cuando los individuos coinciden en un rango sensorial. La evaluación de los individuos en el entorno se realiza a través de la energía, y los reemplazos se producen cuando un robot consume toda su energía o alcanza una vida máxima, lo que produce los reemplazos de los controladores por los embriones. Este algoritmo fue aplicado satisfactoriamente y analizado en varias tareas de coordinación multi-robot, tales como vigilancia o limpieza, y en otras de optimización colectiva como rutas de fletes de buques.

El éxito práctico del algoritmo ASiCO en la coordinación de sistemas multi-robot confirmó la potencialidad del paradigma evolutivo dEE para la resolución de este tipo de casos tan complejos. Pero durante su aplicación, se detectaron también diversos problemas. Los principales han sido los siguientes: tiene una parametrización muy numerosa y además está muy ligada a cada tarea específica, requiere un ajuste manual de estos parámetros y la implementación de los operadores básicos es muy específica de cada tarea. En consecuencia, no se pueden extraer conclusiones generales sobre el funcionamiento del algoritmo porque los resultados no son extrapolables a otras tareas. Esta falta de estandarización es común a los diferentes algoritmos de dEE que existen en la bibliografía. En consecuencia, de la observación de estos problemas comunes, ha surgido la necesidad de diseñar un nuevo algoritmo que no cuente con las mismas limitaciones y que permita valorar el verdadero potencial de los algoritmos de dEE en problemas reales. El diseño, implementación y validación de este nuevo algoritmo, que se ha denominado algoritmo canónico de dEE, es el objetivo principal de esta tesis.

Para diseñarlo, como punto de partida, se han analizado los principales algoritmos de dEE, como son el algoritmo original de Watson, el mEDEA y el ASiCO, con el objetivo de reducir este paradigma a su .esqueleto.eidentificar los procesos básicos que lo conforman, obteniendo una aproximación canónica y general. Además, el algoritmo canónico se ha independizado totalmente del entorno y de la tarea. Tanto las interacciones como los intercambios genéticos vienen determinados ahora por distribuciones de probabilidad y no de interacciones locales en rangos sensoriales. La evaluación es individual y genérica, en la definición del algoritmo no existe el concepto de energía, sino son las tareas concretas las que introducen la necesidad de evaluar introduciendo el concepto de energía.

El algoritmo se ha parametrizado con un número de parámetros mínimo con el fin de que sea fácil de configurar en nuevas tareas. Con objeto de analizar la relevancia de estos parámetros sobre el comportamiento, se ha decidido realizar un análisis de sensibilidad en los espacios de búsqueda de problemas multi-robot a los que nos enfrentamos en esta tesis. Sin embargo, se ha detectado que dentro del campo de la optimización colectiva no existe ningún conjunto de tareas de referencia que permita comparar algoritmos rigurosamente ni se ha estudiado la caracterización de los espacios de búsquedas como se estudiado en otros dominios. Las tareas a las que se enfrentan los algoritmos de optimización colectiva en problemas multi-robot son tradicionalmente simples y muy similares entre sí. Un ejemplo de esto son las tareas de fototaxia (dirigirse hacia la luz), navegación evitando obstáculos o exploración colectiva. Como excepción, existen algunos trabajos con tareas de mayor complejidad, como por ejemplo, tareas de construcción colectiva que requieren sincronización entre los robots para resolverse, pero con una reducida transferencia a la realidad.

En consecuencia, en esta tesis, se ha analizado la bibliografía con el fin de desarrollar un conjunto de funciones sintéticas de referencia que caractericen los tipos de espacio de búsqueda más comunes en problemas colectivos. Una vez definido este conjunto de funciones, se ha analizado la sensibilidad de los parámetros del algoritmo canónico de dEE. Para ello, fue necesario desarrollar una metodología inspirada en el método "innovization", presentado por Deb en 2006, que propone aplicar un algoritmo multiobjetivo para analizar las soluciones en el frente de Pareto en busca de principios de diseño surgidos a través de la optimización, de ahí su nombre (innovaciones a través de optimización), y que no son directamente extraíbles por el diseñador ni intuitivos a priori, sino que es información proporcionada por la propia optimización. El algoritmo multiobjetivo funciona como un meta-algoritmo que evoluciona los parámetros con los que a su vez el algoritmo canónico se ejecuta. Los dos objetivos con los que se aplica el algoritmo multiobjetivo son los siguientes: maximizar el rendimiento del algoritmo canónico (calidad global en la tarea) y minimizar la variabilidad en los resultados (desviaciones de la calidad en la tarea). De este estudio se han concluido rangos óptimos para el funcionamiento de los parámetros según el tipo de espacio de búsqueda o su dificultad.

No solo se pretende que el algoritmo canónico resuelva las tareas de forma óptima, sino también que adapte su solución al tipo de problema, por ejemplo, permitiendo obtener especialización de los individuos en los casos que esto

proporcione una ventaja, o soluciones homogéneas si corresponde. Para ello, en las funciones sintéticas en donde se ha validado el algoritmo canónico, se contemplan funciones con espacios de calidad que requieran especialización. Como se ha comentado anteriormente para los algoritmos CCEAS, la especialización es un mecanismo a estudiar en estos sistemas basados en evolución natural en los que se busca coordinación emergente, ya que permite abordar un problema complejo como un conjunto de sub-problemas más simples. Por este motivo, el estudio de la especialización emergente en el algoritmo canónico ocupa un espacio muy importante en esta tesis, y es una de las principales virtudes de las aproximaciones basadas en EE.

Por último, una vez que se ha analizado el funcionamiento del algoritmo canónico de dEE en un conjunto de funciones sintéticas, se ha estudiado en una tarea inspirada en un problema de vigilancia en interior con drones autónomos. Esta tarea tiene gran relevancia en el marco de esta tesis, ya que contiene todas las características del tipo de problema a resolver en cuanto a complejidad de la solución, posibilidad de optimización mediante una aproximación coordinada y aplicabilidad real. La navegación en interiores es más compleja que en exteriores debido a la restricción de no poder usar sistemas de posicionamiento como GPS, y se debe apoyar en sensores a bordo como cámaras para poder visualizar marcadores naturales o artificiales, y realizar una estimación de la posición a partir de estos. En este problema, la realización de la tarea de vigilancia colectiva depende de resolver previamente una tarea implícita: el problema de la localización. Para dotar del realismo adecuado a la tarea, se ha implementado un entorno en simulación que contiene un modelo de funcionamiento extraído de la caracterización de un dron real, en concreto el modelo PARROT AR.DRONE 2.0. En el simulador se ha incluido el modelo de respuesta real de los sensores y motores del AR.DRONE 2.0 a los elementos presentes en el entorno, fundamentalmente paredes y marcadores artificiales. Desde un punto de vista práctico, se ha comprobado la validez del algoritmo canónico para resolver este tipo de problema de alto grado de realismo en tiempo real y con condiciones cambiantes en la tares. Además desde el punto de vista del desarrollo de un nuevo algoritmo, se ha comprobado la validez de los rangos de los parámetros obtenidos en el caso sintético, y si son relevantes en una tarea real.

El algoritmo canónico de dEE desarrollado se ha probado en esta tarea real contra otros algoritmos similares, en concreto, contra una versión encapsulada de EE y con un algoritmo de CCEA que es referencia en este campo para

la optimización de funciones de alta dimensionalidad. En estas pruebas se ha comprobado que el algoritmo canónico de dEE proporciona un rendimiento práctico mucho mayor que el resto, logrando niveles óptimos de resolución de la tarea en tiempos mucho menores. En un análisis más detallado de las soluciones se ha podido observar que el principal motivo detrás de esta mejora reside en su capacidad para obtener individuos especializados en ciertas sub-tareas de manera automática.

La presente memoria de tesis se ha estructurado de la siguiente forma: en el capítulo 2 se presentarán el objetivo global y los sub-objetivos de la tesis, en el capítulo 3 se revisarán los principales trabajos que abordan la optimización de sistemas multi-robot, con especial énfasis en las aproximaciones evolutivas. En el capítulo 4 se expondrán las especificaciones del sistema multi-robot que se pretende optimizar así como los requisitos de diseño para el algoritmo que los debe lograr, y se justificará la elección del paradigma Embodied Evolution. En el capítulo 5 se presentará en detalle el diseño e implementación del algoritmo canónico de dEE, y la parametrización propuesta para el mismo. En el capítulo 6 se describirá el análisis de sensibilidad realizado sobre el algoritmo canónico y los resultados alcanzados. En el capítulo 7 se describe la tarea real sobre la que se valida el desarrollo y se analizan los resultados obtenidos, tanto a nivel del algoritmo como en comparación con otros algoritmos similares. En el capítulo 7, se extraerán las principales conclusiones del trabajo realizado en esta tesis y en el capítulo 8 se plantearán las principales líneas de trabajo que se abren en el futuro.

Capítulo 2

Objetivos

La motivación original de esta tesis doctoral ha sido la constatación de que los algoritmos evolutivos existentes en la bibliografía poseen una gran potencialidad para la optimización de sistemas multi-robot, dada su gran capacidad de búsqueda en espacios de alta dimensionalidad, pero no se ha logrado aún una aplicabilidad real satisfactoria. La mayor parte de las aproximaciones existentes relajan las especificaciones del sistema final al que se transfieren motivadas por lograr un resultado adecuado, ignorando para ello criterios fundamentales como el tiempo de cómputo, la complejidad de los robots requeridos o la falta de adaptabilidad de la solución. Para satisfacer esta motivación inicial, dentro del Grupo Integrado de Ingeniería de la Universidade da Coruña, se recurrió al paradigma evolutivo Embodied Evolution, surgido a finales de los años 90 para afrontar el diseño automático y en tiempo real de sistemas multi-robot mediante algoritmos evolutivos. En este sentido, se desarrolló dentro del grupo de investigación un algoritmo que sigue este paradigma, denominado ASiCo, y durante años se aplicó a diferentes ámbitos de la optimización colectiva con éxito. Pero su excesiva parametrización y sensibilidad al problema concreto, común a otros algoritmos basados en Embodied Evolution, hicieron que surgiese el objetivo principal tras esta tesis doctoral. Formalmente:

El objetivo principal de esta tesis doctoral es el desarrollo de un algoritmo evolutivo para la optimización de sistemas multi-robot basado en el paradigma Embodied Evolution, que formalice las diferentes variantes que han surgido bajo esta aproximación

Se trata, en definitiva, de comenzar una línea de trabajo ambiciosa en la estandarización de este paradigma con el objetivo de conocer sus fortalezas y debilidades de manera formal a la hora de ser aplicado en el diseño automático de sistemas multi-robot reales. Para lograr este objetivo global se hace necesario completar los siguientes sub-objetivos parciales:

- Analizar los diferentes algoritmos de Embodied Evolution que existen en la literatura y extraer los procesos básicos que caracterizan su funcionamiento.
- Desarrollar un algoritmo canónico que estandarice esta familia de algoritmos. Para ello se debe:
  - Modelar matemáticamente los procesos básicos de forma general
  - Definir la parametrización más simple posible de estos modelos
  - Crear un pseudocódigo que implemente los procesos básicos detectados
- Realizar un análisis del funcionamiento del pseudocódigo propuesto y un análisis de sensibilidad de la parametrización. Para hacer esto es necesario:
  - Proponer un conjunto de funciones sintéticas que representen los diferentes espacios de búsqueda a los que se puede enfrentar un algoritmo de este tipo en problemas colectivos
  - Proponer una tarea real para un sistema multi-robot que cumpla las especificaciones de aplicabilidad deseadas
  - Proponer una metodología experimental para realizar el análisis de funcionamiento y sensibilidad
- Extraer conclusiones sobre los rangos de funcionamiento óptimo de los parámetros en función de las propiedades del espacio de búsqueda
- Comparar el rendimiento del algoritmo frente a otros algoritmos similares en la optimización de sistemas multi-robot
- Realizar una primera aproximación al estudio formal de este tipo de algoritmos basados en EE en cuanto a sus dominios de aplicación, puntos fuertes y debilidades

Capítulo 3

Antecedentes

Un sistema multi-robot se puede definir como un conjunto de robots que operan sobre la misma tarea. Suele ser aplicado para resolver problemas colectivos (aquellos que requieren más de un robot para ser resueltos) y distribuidos, pero también se puede utilizar en problemas que no requieren una solución colectiva, pero en los que resolver la tarea colectivamente produce algún beneficio. De manera general, las principales ventajas de los sistemas multi-robot son las siguientes:

- son capaces de resolver tareas complejas al dividir la tarea global en otras más sencillas
- aumentan la eficiencia en la resolución de la tarea si ésta se puede realizar en paralelo
- son robustos y tolerantes a fallos debido a su redundancia
- son flexibles y escalables. Pueden estar formados con robots de diferentes capacidades (sensores y actuadores) o con las mismas, y pueden ser tolerantes a cambios en el número de integrantes del equipo

Se deben diferenciar los sistemas multi-robot de otros campos relacionados como los sistemas multi-agente o la inteligencia artificial distribuida. En primer lugar, se puede definir un agente como un mecanismo computacional con una alta autonomía que interactúa en un entorno virtual o simulado [Panait and

Luke, 2005]. Los sistemas multi-agente son modelos formados por un conjunto de agentes compartiendo un mismo entorno. En segundo lugar, la inteligencia artificial distribuida afronta problemas complejos y busca soluciones distribuidas mediante nodos de procesado, y se puede considerar un campo precursor de los sistemas multi-agente. En contraste con ambos casos, los sistemas multi-robot se enfrentan a entornos reales, lo cual implica una serie de restricciones y prioridades a la hora de llevar a cabo su optimización. Como explican Russel y Norvig en su libro [Russell and Norvig, 2010], el mundo real es inaccesible (los sensores solo perciben estímulos cercanos al robot), es no determinista (las ruedas derrapan, las piezas se rompen, y un robot debe lidiar con la incertidumbre), no episódico (los efectos de las acciones cambian a lo largo del tiempo), dinámico (se producen cambios fuera del control del propio robot), y continuo (no se pueden enumerar las posibles acciones). Esto ha implicado que la investigación en sistemas multi-agente se haya centrado más en protocolos de interacción, comunicaciones o lenguaje, dejando de lado los problemas derivados de la propia operación del robot en el mundo real. Debido a que la motivación de esta tesis es el desarrollo de un algoritmo para la optimización de sistemas multi-robot, se revisarán trabajos enfocados a éstos, aplicados tanto a robots reales como simulados pero manteniendo un alto grado de realismo en la tarea, y no se describirán trabajos desarrollados para sistemas multi-agente. De todas formas, hay que destacar que en muchos trabajos en sistemas multi-robot se suelen relajar las condiciones del mundo real, que resulta en simulaciones en las que la diferencia entre sistema multi-robot y sistema multi-agente no se percibe con facilidad.

La presente tesis doctoral se centra en la optimización de los sistemas multi-robot. Dicha optimización se lleva a cabo en dos niveles:

1. Optimización de la morfología: centrada en la obtención de la composición ideal del equipo a nivel hardware, tanto en el número de robots como en sus características en cuanto a sensores y actuadores, es decir, sus capacidades de operación. La optimización de la morfología es un tema poco tratado el la bibliografía, pero muy relevante a la hora de lograr un sistema realmente optimizado.

2. Optimización del control: centrada en el la obtención del control de cada robot individual, con el objetivo de lograr un comportamiento coordinado óptimo. La mayor parte de los trabajos que se revisarán, y también el

principal esfuerzo de desarrollo de esta tesis, están en este nivel.

El presente capítulo se ha organizado de la siguiente manera. Inicialmente se realizará una clasificación de los sistema multi-robot en base a sus principales características, de modo que quede claro sobre qué elementos actúan los dos niveles de optimización comentados con anterioridad. A continuación, en el apartado 3.2, se revisarán las principales aproximaciones a la optimización de sistemas multi-robot basadas en algoritmos evolutivos, que es el principal campo de interés de esta tesis.

3.1. Clasificación de sistemas multi-robot

Existen diversas características por las que pueden clasificar los sistemas multi-robot, y ha habido en la literatura diferentes esfuerzos por hacerlo, aunque todas las clasificaciones coinciden en algunos criterios comunes. Una de ellas es la proporcionada por Farinelli [Farinelli et al., 2004], en la que se clasifican los sistemas multi-robot por tipo de coordinación (cooperar o competir), nivel de conocimiento de cada robot (individual o colectivo) y tipo de organización (fuertemente centralizada, débilmente centralizada, o distribuida). Otra clasificación diferente aparece en el trabajo de Duro et al. [Duro et al., 2010], que se basa en las siguientes características: acoplamiento, morfología, entorno, control y percepción. En cuanto al acoplamiento, los robots pueden presentan acoplamiento físico, como los s-bot [Dorigo et al., 2013], acoplamiento funcional, dependiendo de otros robots para realizar una acción, (por ejemplo, colaborar entre varios para pasar un obstáculo), o acoplamiento de información, que exista comunicación entre ellos o no. En cuanto a la morfología, los autores distinguen entre sistemas heterogéneos u homogéneos. Relativo al entorno, los autores comentan que no es estrictamente una característica del sistema multi-robot pero sí condiciona su operación, y también su diseño. En relación con el control, diferencian entre un control centralizado o control distribuido. Finalmente, en cuanto a la percepción, la capacidad de percepción de cada uno de los robots determina la información con la puede trabajar su controlador. Es posible que para realizar una tarea, como por ejemplo la creación de un mapa del entorno, se necesite fusionar las sensorizaciones de los diferentes robots del sistema. Otra taxonomía es la de Cao et al. [Cao et al., 1997], enfocada en sistemas multi-robot cooperativos, y en la que se clasifican los sistemas en base a tres criterios:

arquitectura de grupo, resolución de conflictos, y origen de la cooperación. El primer criterio, la arquitectura de grupo, se refiere a diferentes aspectos como la centralización o descentralización, la diferenciación entre robots homogéneos o heterogéneos, la estructura de comunicación o la modelización o no de otros robots por parte de un robot. El segundo criterio se refiere a la forma de resolver los conflictos por recursos, y por último, el tercer criterio se centra en cómo se produce la cooperación. Otra taxonomía relevante es la de Dudek et al. [Dudek et al., 2002]. Los criterios de esta taxonomía son el número de robots en el sistema (1, 2, o n), el rango de comunicaciones de los robots (distancia a la que pueden comunicarse), la topología de las comunicaciones (con qué robots pueden comunicarse), el ancho de banda de las comunicaciones (cuánta información se pueden transmitir), la reconfigurabilidad (la frecuencia con la que pueden modificar su organización), la capacidad de procesado y la composición colectiva (si son robots heterogéneos o robots homogéneos).

Como se puede observar, no existe un criterio homogéneo para seleccionar las características que definen los sistemas multi-robot. Lo que sí parece evidente es la existencia de una cierta mezcla entre características hardware y de control, que creemos que deben ser claramente diferenciadas. En esta tesis, a partir de la revisión anterior, se propone una nueva taxonomía más simple y que está basada en los siguientes cuatro criterios: morfología, arquitectura de control, comunicación y coordinación.

3.1.1. Morfología

Por morfología se entiende el diseño físico de los robots, fundamentalmente su forma, sensores y actuadores, lo cual determina sus capacidades operativas. En un primer nivel, se puede diferenciar entre sistemas multi-robot heterogéneos, formados por robots con morfología diferente, y equipos de robots homogéneos, formados por robots morfológicamente idénticos. Estos últimos tienen la ventaja de que los robots son fácilmente intercambiables, pueden llevar a cabo todas las funciones y tienen mayor tolerancia a fallos. La desventaja es que, en general, pueden ser robots con hardware más complejo para poder realizar cualquier tipo de operación. Los sistemas con morfología heterogénea tienen la ventaja de ahorrar costes con robots con capacidades específicas a costa de perder tolerancia a fallos y flexibilidad.

Tradicionalmente, la morfología de los robots ha sido una especificación pre-

fijada por el diseñador, es decir, típicamente se parte de un equipo de robots predefinido en número y capacidades, y se busca optimizar el control del mismo. La selección de la morfología suele atender a criterios prácticos, en cuanto a que las capacidades de los robots permitan desarrollar la tarea, y también en cuanto al propio coste de adquisición del sistema, que no es un criterio despreciable. Comenzaremos por revisar algunos ejemplos relevantes de sistemas multi-robot tanto homogéneos como heterogéneos, todos ellos partiendo de diseños predefinidos.

Un ejemplo clásico de sistema multi-robot con hardware homogéneo son los enjambres de robots [Trianni, 2008] [Brambilla et al., 2013], que utilizan un número alto de robots idénticos con capacidades de sensorización y actuación simples. Los enjambres de robots están inspirados en sistemas colectivos de la naturaleza como los insectos sociales, colonias de hormigas, bandadas de pájaros o bancos de peces. La robótica de enjambre (Swarm Robotics) constituye una rama en sí misma, que no solo utiliza los sistemas naturales como inspiración a la hora de definir la morfología de los robots, sino también a la hora de diseñar su control, como comentaremos más adelante. Otro ejemplo popular de sistema multi-robot homogéneo son los utilizados en las competiciones de futbol con robots, como la Robocup [Kitano et al., 1997], en las que se enfrentan dos equipos con unas reglas y un terreno de juego adaptados. Estas competiciones se realizan cada año y existen diferentes categorías según los robots usados, que son siempre homogéneos. Hasta el 2004, la plataforma estándar de la Robocup era el robot Aibo, un cuadrúpedo fabricado por Sony. A partir de 2008, la plataforma estándar de esta competición ha sido el robot humanoide Nao. Otro tipo de sistema con morfología homogénea y de gran aplicabilidad futura son los nanorobots, sistemas microscópicos que serán aplicados al ámbito de la medicina, que operarán dentro del cuerpo de un paciente y podrán aplicar cirugías o transmitir medicamentos [Mallouk and Sen, 2009].

En cuanto a sistemas con morfología heterogénea, se pueden destacar los ejemplos prácticos realizados por Simmons et al. [Simmons et al., 2001], en el que coordinan un equipo de tres robots con diferentes capacidades para realizar una tarea de ensamblaje. Uno de los robot tiene una grúa con lo que es capaz de levantar objetos pesados, otro robot es un manipulador móvil que es más preciso pero no puede manipular objetos pesados y un tercer robot que se encarga de observar el proceso y guiar a los otros robots.

Otro ejemplo de utilización exitosa de equipos heterogéneos es el framework

MURDOCH [Gerkey and Mataric, 2002], que coordina el sistema a través del concepto de subastas, que será explicado más adelante con detalle. En este caso, los robots utilizados son Pioneer 2-DX que pueden ser configurados con diferentes accesorios: cámaras, sónares, láseres. El equipo estaba formado por: tres robots con cámaras, tres robots con cámaras y láseres, un robot con una cámara y sensores de contacto, y un ordenador con una cámara. Un ejemplo más reciente de sistema heterogéneo aparece en el proyecto Swarmanoid [Dorigo et al., 2013], centrado en la obtención del control automático de un sistema multi-robot con capacidades diferentes. En este caso, se utilizan tres tipos de robots diferentes con características y capacidades complementarias, de forma que ofrecen una variedad de capacidades mayor que las que podría realizar un solo robot. En concreto, se utilizan tres robots con ruedas y tracción de oruga que se pueden acoplar entre sí, un robot manipulador que puede agarrar objetos pero no se puede mover, y un robot volador que proveen de una vista aérea para localizar objetos a otros robots.

Como se puede extraer de los trabajos comentados anteriormente, el sistema de control del equipo multi-robot debe soportar el tipo de morfología hardware del sistema. El tipo de sistema más general es el heterogéneo ya que no limita a un único diseño de robot pero complica la optimización debido a que los controladores no son trasladables de un robot a otro y el controlador de cada de uno de ellos debe aprovechar todas las capacidades.

En cuanto a la optimización de la morfología, los principales trabajos en esta línea se enmarcan dentro de la robótica modular, un sub-campo de los sistemas multi-robot que se basa en que los robots están formados por módulos independientes que se pueden configurar de formas diferentes, dando lugar a morfologías hardware adaptadas a la tarea. Dos proyectos europeos en esta línea que se deben destacar, son los proyectos Replicator y Symbrion [Levi and Kernbach, 2010]. Ambos se centran en el concepto de organismo multi-robot, en el cual los robots son capaces de agregarse y desagregarse con otros, y los controladores funcionan tanto a nivel individual como a nivel agregado. El proyecto Symbrion se centra aplicar evolución artificial para obtener estos organismos.

3.1.2. Arquitectura de control

La arquitectura de control de un sistema multi-robot es la encargada de establecer cómo se gestiona el control del equipo en cuanto a la independencia de cada robot para tomar sus propias decisiones manteniendo el objetivo global de la tarea colectiva. La decisión de cómo se gestiona dicho control es de gran relevancia a la hora de que el sistema sea trasladable a un sistema real. Un trabajo de gran relevancia en donde se proponen los diferentes tipos de arquitecturas de control posibles, es la revisión realizada por Parker [Parker, 2008], que distingue entre:

- Centralizadas: se basan en la existencia de un punto central de control, con un robot que ejerce de líder. Tiene la gran desventaja de que se introduce un punto de fallo crítico en el sistema y la dificultad de comunicar todos los robots a este punto central con independencia de su localización y circunstancias. Un ejemplo de este tipo de arquitectura es la arquitectura MARTHA [Alami et al., 1998] o la arquitectura de Milutinovic [Milutinovic and Lima, 2006].
- Jerárquicas: cada robot supervisa las acciones de un conjunto reducido de robots, y así sucesivamente, hasta el nivel más básico, en el que un robot individual realiza la tarea sin supervisar a ningún otro. Un trabajo reciente que sigue esta arquitectura es el trabajo de Chand et al. [Chand and Carnegie, 2013], o el trabajo de Grabowski et al. [Grabowski et al., 2000].
- Distribuidas o descentralizadas: los robots realizan el control basándose en información local sin que existan comunicaciones centralizadas. Poseen mayor robustez y pueden continuar resolviendo el problema a pesar de que uno o varios robots dejen de funcionar. También permiten utilizar robots con sensores y actuadores más simples. La principal desventaja que presentan que los sistemas distribuidos es que su control coordinado es más difícil de diseñar, ya que se basa en la información local de cada uno. Un ejemplo de este tipo de arquitectura es ALLIANCE [Parker, 1998] o la arquitectura Embodied Evolution de Watson et al. [Watson et al., 2002], que se tratará más adelante con detalle.
- Híbridas: se combina el control basado en información local con control de tipo centralizado que utiliza información global. Un ejemplo es la arqui-

tectura Distributed Robot Architecture (DIRA) [Simmons et al., 2001] o la arquitectura de Tanner et al. [Tanner and Kumar, 2005].

Como se puede observar, existen diferentes aproximaciones al problema de cómo gestionar el control coordinado que van desde aquellos que buscan un mayor conocimiento sobre la coherencia del sistema y una menor complejidad (centralizados), a aquellos que dan gran libertad a los robots individuales buscando una mayor robustez (distribuidos). La optimización del sistema multi-robot se ve condicionada por el tipo de arquitectura seleccionada, aunque no hemos encontrado ningún trabajo en la bibliografía en el que se optimice el tipo de arquitectura a utilizar, es decir, que si la aproximación es más centralizada o distribuida ha sido siempre una decisión de diseño previa, a partir de la cual se realiza la optimización del sistema.

3.1.3. Comunicación

Como se comentó anteriormente, el entorno real en el que se desenvuelve un sistema multi-robot es inaccesible, de modo que cada robot solo tiene acceso a la información del mundo que le rodea y que puede captar a través de sus sensores. Esta limitación se puede resolver de manera simple transmitiendo información entre los robots, de manera que todos posean información global relevante para la resolución de la tarea colectiva. En relación con la arquitectura de control, la comunicación de la información puede ser centralizada o distribuida, y como se puede suponer, la complejidad de gestionar una u otra es muy diferente. En un sistema centralizado, el líder mantiene la información global actualizada a partir de la que recibe de cada robot individual, y la comparte con el resto del robots del equipo. En los sistemas distribuidos, por otro lado, los robots comparten entre sí la información que poseen (a diferentes niveles en función del caso), de modo que no se puede garantizar que todos los robots posean información global completa en cada instante, pero sí un conocimiento mayor que el individual puro.

Independientemente de si la información está centralizada o distribuida, podemos distinguir dos tipos de sistemas multi-robot en cuanto al tipo de comunicación utilizado. Por un lado, aquellos que emplean comunicación explícita, en la que los robots se comunican directamente entre sí, y por otro lado, los que emplean comunicación implícita, que es un tipo de comunicación a través

de la modificación del entorno conocida como "estigmergia". Este tipo de comunicación está inspirado en los sistemas naturales, por ejemplo, en el caso de la comunicación entre hormigas, que dejan un rastro de feromonas a su paso que otras hormigas pueden detectar y seguir [Holland and Melhuish, 1999]. Los trabajos con enjambres de robots utilizan típicamente comunicación implícita, como por ejemplo [Schneider-Fontan and Mataric, 1998], que resuelve una tarea de limpieza por áreas con "estigmergia". Otro ejemplo en este sentido es el trabajo de Kube et al. en el que unos robots deben empujar una caja sin comunicación explícita [Kube and Zhang, 1993]. Por otro lado, existen numerosos ejemplos de sistemas multi-robot con comunicación explícita, como son los sistemas basados en subastas entre robots [Gerkey and Mataric, 2002] [Zlot and Stentz, 2006], y que se detallarán en el siguiente sub-apartado.

De nuevo, no se han encontrado trabajos que consideren el tipo de comunicación (explícita o implícita) como una característica a optimizar, sino que siempre se ha predefinido dicha capacidad en función del hardware disponible en los robots (las comunicaciones requieren elementos que las posibiliten), y de la complejidad de la tarea.

3.1.4. Coordinación

La definición de coordinar según la RAE es "unir dos o más cosas de manera que formen una unidad o un conjunto armonioso". Para que un sistema multi-robot realice un comportamiento colectivo es necesario que exista una coordinación entre los robots que forman el sistema, produciendo que el comportamiento colectivo sea más eficiente que la simple agregación de comportamientos individuales. De hecho, la ejecución independiente de comportamientos en los robots, es decir, sin ningún tipo de coordinación, no se considera de interés en la optimización de sistemas multi-robot y no será tratada en esta tesis.

En una primera aproximación, la coordinación sistemas multi-robot se puede dividir en dos tipos: cooperativa o competitiva. En sistemas cooperativos, los robots persiguen un objetivo común en beneficio del equipo, mientras que en sistemas competitivos, los robots persiguen objetivos en beneficio individual y en detrimento de otros robots, pero que pueden dar lugar a la resolución colectiva de una tarea. Por ejemplo, los sistemas multi-robot basados en subastas en los que los robots compiten por ganar las pujas y realizar alguna tarea en su propio beneficio individual [Zlot and Stentz, 2006], se pueden diseñar de

manera que estas pujas lleven a que los robots cubran la totalidad de tareas individuales que forman una tarea global compleja. Ejemplo de sistemas con coordinación cooperativa son mucho más comunes, por ejemplo, en un trabajo de Matarić et al. en el que dos robots Genghis II tienen que empujar una caja cooperativamente [Mataric et al., 1995].

Además de esta distinción inicial, la coordinación de sistemas multi-robot se puede organizar en intencional o emergente, en función de si las tareas que realiza cada robot están predefinidas a priori o no. En sistemas con coordinación intencional, el problema es cómo asignar las tareas a cada uno de los robots para aumentar el rendimiento del sistema (problema de asignación de tareas a sistemas multi-robot o MRTA). Para profundizar en este tipo de coordinación, una revisión completa de los trabajos existentes se puede encontrar en la taxonomía presentada por Gerkey y Matarić [Gerkey and Matarić, 2004]. En este trabajo se describen, en primer lugar, los diferentes tipos de problemas MRTA y se clasifican en base a tres criterios: si los robots pueden hacer una tarea o varias simultáneamente, si las tareas requieren a un robot o a varios y si es una asignación instantánea o es una asignación que se recalculará de forma iterativa a medida que se modifican las condiciones de la tarea. Típicamente, en los sistemas con coordinación intencional se debe optimizar un valor de utilidad que se calcula a partir del coste de las tareas para cada robot y la calidad obtenida por realizar la tarea. Dentro de las arquitecturas de coordinación intencional, se debe destacar la arquitectura ALLIANCE [Parker, 1998], una de las primeras arquitecturas de asignación de tareas que utiliza robot reales. Otra arquitectura relevante es la BLE (Broadcast of Local Eligibility) [Werger and Matarić, 2000], que está basada en la arquitectura de subsunción de Brooks [Brooks, 1986] y que utiliza un algoritmo voraz para la asignación. También se deben destacar trabajos basados en aspectos económicos como la arquitectura M+ de Botehlo y Alami [Botelho and Alami, 1999], que fue la primera aproximación basada en mercados aplicada a sistemas multi-robot, y que se basa en negociaciones entre los robots para asignarse tareas. Otras arquitecturas basadas en mercados son la arquitectura MURDOCH [Gerkey and Mataric, 2002], la arquitectura TraderBots [Zlot and Stentz, 2006] y la arquitectura Hoplites [Kalra et al., 2005]. Las dos primeras están basada en un modelo de subastas en el que los robots reciben una recompensa por realizar una tarea con un coste asociado, y en la que el objetivo es maximizar el beneficio de todo el sistema. Por otro lado, en la arquitectura Hoplites [Kalra et al., 2005] se selecciona un plan conjunto que tiene en cuenta los beneficios y el coste de ese plan.

Una ventaja de la coordinación intencional es que tiene el potencial de aprovechar mejor las capacidades en un sistema con hardware heterogéneo, y también que son sistemas con una aplicabilidad más directa a problemas conocidos, ya que las tareas vienen predefinidas por el diseñador. Esta es, a su vez, su principal desventaja en casos más generales y reales, donde no se tiene un control absoluto del entorno y la definición previa de tareas es compleja.

En cuanto a sistemas multi-robot con coordinación emergente, su principal fuente de inspiración ha sido la coordinación que se observa en sistemas colectivos naturales. La principal ventaja con respecto la coordinación intencional, es que, como se ha dicho, no es necesario predefinir las tareas que realizará cada robot, sino que surgen de la propia interacción entre ellos y del objetivo final a resolver. La desventaja principal es que obtener el control coordinado de esta forma es un problema mucho más complejo, ya que existen infinidad de combinaciones posibles de comportamientos individuales que pueden dar lugar al mismo comportamiento colectivo, por lo que los grados de libertad del espacio de búsqueda se incrementan en gran medida. De hecho, las tareas individuales surgen de manera automática, y es el observador externo el que las define a posteriori.

Un tipo de sistema multi-robot representativo que emplea coordinación emergente son los enjambres de robots, que como se dijo anteriormente, constituyen un campo de desarrollo propio llamado robótica de enjambre. La robótica de enjambre estudia mecanismos de auto-organización y se centra en la coordinación a partir de interacciones locales de un número relativamente alto de robots (con respecto otros sistemas multi-robot). Se pueden diferencian los enjambres respecto otros sistemas multi-robot por sus características propias: son siempre cooperativos, totalmente distribuidos y las interacciones tienen que ser locales [Brambilla et al., 2013], incluyendo las comunicaciones. Los métodos que se emplean para diseñar el control en enjambres suelen estar basados en comportamientos predefinidos. Así, es usual que los enjambres estén basados en reglas, en los que se diseña el sistema de abajo a arriba [Crespi et al., 2008], es decir, predefiniendo unas reglas de interacción entre los robots que producen el comportamiento global deseado, como por ejemplo, modelando la quimiotaxis de la bacteria E.coli para resolver una tarea de búsqueda con robots e-puck [Pugh and Martinoli, 2008]. Otra aproximación que produce coordinación emergente pero sin reglas predefinidas son las máquinas de estado finitas probabilísticas, en las que los robots basan su acción según su estado y sus sensorizaciones. Los

robots realizan transiciones entre posibles estados según valores de umbral y realizan la acción que corresponde a cada estado. Algunos ejemplos de tareas solucionadas con esta aproximación son una tarea de búsqueda de comida [Liu et al., 2007] o una tarea de agregación [Soysal and Sahin, 2005]. Finalmente, otro método de coordinación emergente utilizado en enjambres de robots son las reglas físicas virtuales, en donde se aplican campos de fuerza que atraen o repelen a los robots. Estos métodos son muy apropiados para realizar tareas de formación con los robots. Un ejemplo de estos métodos es el framework "Physicomimetics" [Spears et al., 2004] o un trabajo de Spears et al. [Spears et al., 2006].

Otro tipo de aproximación muy relevante para obtener coordinación emergente en sistemas multi-robot, tanto de enjambre como de tipo general, son las técnicas de diseño automático, que se fundamentan en no predeterminar ningún tipo de regla o comportamiento en el sistema multi-robot, y que constituyen la aproximación que seguiremos en esta tesis. Dentro de las técnicas de diseño automático destacan el aprendizaje por refuerzo y los algoritmos evolutivos. Las dos técnicas se han aplicado mayoritariamente a sistemas de un solo robot, pero también se han adaptado para sistemas multi-robot, afrontando los desafíos que esto supone. Además, ambas técnicas podrían aplicarse simultáneamente con el fin de complementarse entre sí, ya que actúan en diferentes niveles. A continuación, se describirá el aprendizaje por refuerzo aplicado a sistemas multi-robot con mayor detenimiento.

En aprendizaje por refuerzo, los robots perciben el estado del entorno, realizan una acción y obtienen recompensas o costes según unas valoraciones establecidas en la tarea [Sutton and Barto, 1998]. El objetivo de un algoritmo de aprendizaje por refuerzo es maximizar las recompensas y minimizar los costes. De esta forma se acaban asociando los estados con las acciones que deben realizar para completar la tarea. Uno de los algoritmos mas relevantes de aprendizaje por refuerzo es el algoritmo Q-Learning [Watkins and Dayan, 1992]. Su convergencia está garantizada en un entorno estacionario como es el caso de un solo robot. En el caso de múltiples robots, debido a la acción de los otros robots, no se cumple esta condición de entorno estacionario, y por lo tanto, no está garantizada. Existen dos grandes tendencias a la hora de aplicar el algoritmo Q-Learning a un sistema multi-robot. Por una parte, se puede considerar el aprendizaje de cada robot de forma independiente, ignorando las acciones del resto de robots (Independent Learners) [Matarić, 2001], o considerar el apren-

dizaje de las acciones conjuntas (Joint Action Learners) [Claus and Boutilier, 1998], [Littman, 2001]. Al considerar las acciones conjuntas, debido a que cada agente tiene múltiples acciones posibles, el espacio de acciones crece exponencialmente con el número de agentes, lo cual puede complicar la resolución de problemas complejos con muchos estados.

Un tipo de problema colectivo que ha sido un dominio muy estudiado en aprendizaje por refuerzo debido a su dificultad y su amplio espacio de estados, son los problemas basados en fútbol con robots. En este dominio, originalmente planteado por la Robocup [Kitano et al., 1997], se pueden plantear diferentes tareas tanto cooperativas como competitivas, como por ejemplo, mantener la posesión, marcar goles, robar la pelota, defender y evitar goles. Debido a la dificultad de las acciones en estos dominios, para realizar estas tareas se proporcionan a priori una serie de comportamientos predefinidos que el robot tendrá que seleccionar. Un trabajo concreto en este ámbito es el de Asada y Uchibe [Asada et al., 1999], en el que dos robots colaboran para marcar un gol en una portería. Se asumen dos roles: un robot es el pasador y otro es el tirador. El objetivo es que el pasador aprenda a pasar la pelota al otro robot, y el tirador aprenda a tirar a puerta para marcar un gol. Otro escenario posible dentro del mismo dominio de fútbol, en este caso como una tarea competitiva, es la tarea de mantener la pelota, en la que un equipo intenta mantener la posesión mientras el otro equipo intenta robarla [Stone et al., 2005]. En este trabajo se ha usado un algoritmo Sarsa(λ), basado en diferencias temporales. Para esta tarea, al igual que en otros ejemplos de fútbol, los robots tienen una serie de comportamientos de alto nivel: mantener la pelota, pasarla, moverse al hueco, interceptar la pelota, y bloquear una línea de pase entre dos robots. Otro ejemplo de fútbol en un sistema multi-robot es el trabajo de Kok et al. [Kok et al., 2005b], que describe una aproximación que resultó ganadora de la competición Robocup de robots virtuales de 2003. En este trabajo se emplean grafos de coordinación, que son una representación que parte de la premisa de que la función de recompensa global puede ser modelada como una combinación lineal de funciones de recompensa locales que implican coordinaciones locales entre algunos robots.

Otro ejemplo en este dominio es el trabajo de Matarić, en el que un grupo de cuatro robots aprenden a realizar una tarea de búsqueda colectiva de comida con varias aproximaciones basadas en Q-Learning [Matarić, 1997]. Los comportamientos fijos definidos a priori eran los siguientes: moverse de forma

segura sin colisionar, dispersarse, permanecer en reposo, y volver a su base. El problema típico de depredador-presa también se ha abordado en aprendizaje por refuerzo, como por ejemplo, en [Kok et al., 2005a], donde dos depredadores deben coordinarse para cazar a una presa. El entorno propuesto es toroidal en forma de rejilla de celdas en las que, en cada paso de tiempo, tanto depredadores como presa se pueden quedar quietos o moverse a una celda adyacente. El comportamiento de la presa es predeterminado y la tarea concluye cuando los dos depredadores se sitúan adyacentes a la presa.

En la actualidad, se sigue aplicando aprendizaje por refuerzo a la optimización en sistemas multi-robot, aunque de forma menor a lo que fue en años anteriores, y mayoritariamente en combinación con otros métodos de aprendizaje, como por ejemplo, los algoritmos evolutivos que describiremos a continuación.

Los algoritmos evolutivos son técnicas de aprendizaje no supervisado que surgen de aplicar los principios de la evolución natural, principalmente la supervivencia de los individuos más adaptados, a sistemas artificiales. En la evolución artificial se parte de una población de soluciones candidatas que se van modificando y son evaluadas en base a un criterio, la función de calidad, hasta que se obtiene una solución óptima o se cumple una medida de rendimiento. La aplicación de algoritmos evolutivos a la robótica se denomina robótica evolutiva y es un campo que se ha desarrollado, sobre todo, desde la década de los noventa en donde ganó popularidad como un método muy prometedor [Nolfi and Floreano, 2000]. Dentro de la robótica evolutiva, los primeros trabajos abordaban el problema de diseñar un controlador de un solo robot, pero a medida que este campo se desarrollaba, también se puso el interés en sistemas multi-robot y en mecanismos de autoorganización y coordinación. Debido a que el algoritmo desarrollado en esta tesis se enmarca en el campo de los algoritmos evolutivos, a continuación se describirán en más detalle las principales aproximaciones a la optimización multi-robot mediante algoritmos evolutivos.

3.2. Optimización de sistemas multi-robot mediante algoritmos evolutivos

El campo de desarrollo de esta tesis doctoral es la computación evolutiva aplicada a los sistemas multi-robot. Se escapa, pues, del alcance de la misma la revisión del campo de la computación evolutiva fuera de este nicho de aplicación,

dada su extensión y generalidad [De Jong, 2006]. Se debe destacar, eso sí, que la investigación en técnicas basadas en computación evolutiva sigue creciendo desde su aparición y su grado de aplicación en problemas reales va en aumento. De todas formas, con el objetivo de definir la nomenclatura que se usará en posteriores apartados, en el siguiente sub-apartado se realizará una breve explicación del funcionamiento de un algoritmo evolutivo general aplicado a robótica, y también de las particularidades al ser aplicado a sistemas multi-robot.

3.2.1. Esquema de funcionamiento de un algoritmo evolutivo en sistemas multi-robot

El esquema de funcionamiento de una algoritmo evolutivo aplicado a un solo robot se muestra en el pseudocódigo 1. En primer lugar, es necesario codificar los parámetros a evolucionar en un genotipo, que dependiendo del algoritmo utilizado podrían tener una estructura diferente, desde estructuras de árbol hasta vectores de valores reales. En cuanto a qué parámetros codifican la optimización de un sistema mono-robot, el caso es equivalente a un sistema multi-robot, pudiéndose codificar los parámetros del controlador y/o la morfología del robot. A continuación, se crea una población inicial de estos genotipos de forma aleatoria y se comienza un proceso en bucle, realizado en generaciones sucesivas, que mejoran la población anterior. En cada generación, se evalúan los individuos a través de la función de calidad, que debe ser definida por el diseñador a priori. Esta función de calidad es una medida cuantitativa del rendimiento del robot en la tarea, ya sea directa o indirecta. Previamente a que la calidad se pueda calcular, el genotipo se transforma en un fenotipo que es el que realmente recibe la valoración. Las técnicas de evaluación en robótica evolutiva son muchas y muy diversas, pero de modo general, suelen requerir la ejecución del controlador del robot en un entorno real o simulado. Para una revisión exhaustiva de las diferentes opciones de cálculo de la calidad en robótica evolutiva, se recomienda la lectura del trabajo de Nelson et al. [Nelson et al., 2009].

Continuando con el pseudocódigo 1, una vez que se han evaluado todos los individuos, se aplican los operadores típicos de selección y recombinación (cruce y mutación habitualmente). Existen numerosas opciones de operadores que han sido aplicados a sistemas robóticos con éxito [Nolfi and Floreano, 2000], tanto como a otro tipo de sistemas. De todas formas, se debe destacar que los operadores están estrechamente relacionados con la codificación de los genotipos,

por ejemplo, si los genotipos tienen estructura de árbol, los operadores deben actuar sobre estas estructuras tratando de lograr la mayor variabilidad posible de forma que se llegue a una solución óptima. Una vez que se han generado nuevos individuos tras las operaciones de selección y recombinación, se reemplazan los individuos con peor calidad por los nuevos, como es usual. Una vez reemplazados, se comienza una nueva generación y se repite este proceso.

El esquema de funcionamiento para optimizar un sistema multi-robot mediante un algoritmo evolutivo se muestra en el pseudocódigo 2. Como se observa, la codificación es el primer aspecto que puede ser diferente con respecto al caso para un robot, ya que aparecen nuevas opciones. En lugar de codificar a un solo individuo por genotipo, se podría codificar a todo el equipo, o parte de él. Asimismo, en lugar de evolucionar en una sola población, se podría crear una población para cada robot. Estas decisiones son muy relevantes a la hora de lograr una solución más o menos especializada, como se verá más adelante. A la hora de evaluar los individuos, el funcionamiento cambia sustancialmente. Por un lado, si los genotipos codifican a un único individuo, hay que formar combinaciones entre varios robots para crear un equipo que se evalúe conjuntamente en la tarea. Existen muchas opciones para realizar esta evaluación, como por ejemplo, clonar un controlador en todos los robots. A pesar de ser una evaluación conjunta, los individuos tendrían una calidad individual asignada, que es lo que guiaría al algoritmo. A continuación, se aplicarían los operadores genéticos de manera similar al caso mono-robot, aunque ahora se pueden aplicar sobre un individuos o varios, de forma centralizada o distribuida. Finalmente se crearía una nueva generación de individuos, para repetir el proceso hasta un límite de generaciones.

La evolución en sistemas colectivos implica diferenciar dos tipos de calidad: la privada o individual y la global. La calidad privada o individual es la que va asociada al comportamiento de cada robot por separado, y mide su aportación a la tarea, independientemente del resto de robots. La calidad global es relativa al rendimiento colectivo como equipo del sistema multi-robot en la tarea. La evolución se puede guiar por cualquiera de estas dos calidades, pero si lo hace por la privada o individual, ésta debe ir alineada con la global, de forma que si se maximizan las calidades privadas se maximiza la calidad global. Es un problema abierto el cómo diseñar estas funciones para valorar la aportación de un individuo en un sistemas colectivo [Agogino and Tumer, 2008].

Un esquema clásico, a modo de ejemplo, de un algoritmo evolutivo aplica-

Algorithm 1 Esquema de un algoritmo evolutivo aplicado a la optimización de un solo robot

```
codificar en el genotipo parámetros del robot
generar población de genotipos
loop
  for all genotipo en la población do
    transformar genotipo a fenotipo
    evaluar con la función de calidad en la tarea
  end for
  seleccionar individuos para aplicar operadores genéticos
  generar individuos nuevos con los operadores genéticos {nuevos genotipos}
  reemplazar individuos de menor calidad
end loop
transferir mejor controlador al robot real
```

Algorithm 2 Esquema de un algoritmo evolutivo aplicado a la optimización de un sistema multi-robot

```
codificar en el genotipo parámetros de un robot o varios
inicializar población o varias poblaciones
loop
  for all genotipo en la población do
    formar equipo para evaluar
    evaluar equipo y asignar calidad al genotipo
  end for
  generar individuos nuevos
  reemplazar individuos de menor calidad
end loop
```

do a un sistema multi-robot, es el aplicado por Trianni y Nolfi [Trianni and Nolfi, 2009], que se muestra en la figura 3.2.1. El sistema multi-robot está formado por robots homogéneos, tanto a nivel de morfología como de control. En este caso únicamente se optimiza el control de los robots y no su morfología, utilizando para ello una codificación individual mediante una red de neuronas artificiales. Así, cada vez que se evalúa un controlador, este se clona en todos los robots del equipo y se deja operando en la tarea durante un tiempo prefijado. Siguiendo este esquema, los autores son capaces de obtener comportamientos auto-organizativos en un problema de movimiento coordinado en el que los robots tenían que sincronizarse para hacer un movimiento a la vez.

En el siguiente sub-apartado se realizará una revisión de las principales apro-

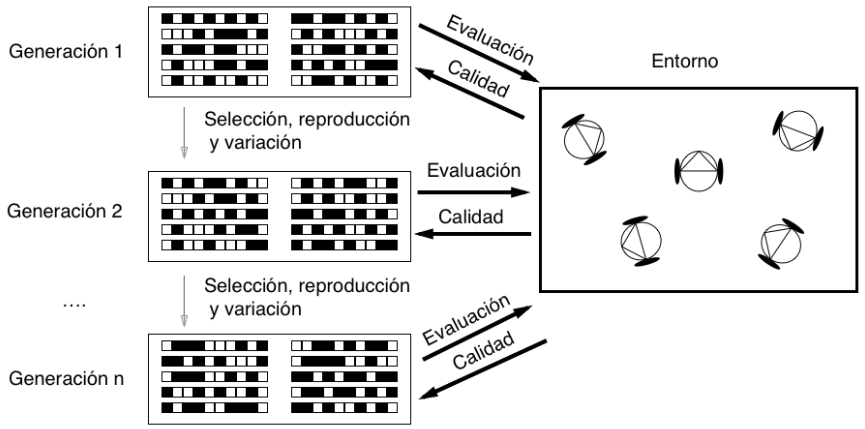


Figura 3.1: Esquema de funcionamiento de un algoritmo evolutivo aplicado a un sistema multi-robot. Fuente: "Evolving collective control, cooperation and distributed cognition" [Trianni and Nolfi, 2009]

ximaciones a la optimización de sistemas multi-robot mediante algoritmos evolutivos, organizando las mismas en una serie de criterios que se han detectado como los más relevantes.

3.2.2. Clasificación de las aproximaciones evolutivas

El campo de la computación evolutiva aplicada a sistemas multi-robot es extenso y bastante heterogéneo, no existiendo hasta la fecha una revisión del mismo que analice en profundidad las diferentes aproximaciones de manera general. Se puede partir de la clasificación propuesta por Ficici y Watson [Watson et al., 2002], basada en dónde y cómo se produce la evaluación de los individuos. En esta clasificación se divide entre aproximaciones que evalúan los controladores en simulación y que son centralizadas, porque mantienen la población desde un punto central, y por otra parte, las aproximaciones en las que las evaluaciones están incrustadas en los robots reales. Las aproximaciones del tipo centralizado requieren mucho tiempo para ser completadas (porque se evalúan los controladores de forma secuencial) mientras que las evaluaciones incrustadas ocurren en tiempo real. Dentro de las aproximaciones de evaluación incrustada, se diferencia entre evaluaciones en serie, cambiando los controladores de forma

secuencial, y evaluaciones en paralelo, en donde cada robot está evaluando un genotipo. Por último, dentro de las aproximaciones que evalúan en paralelo, se distingue entre aproximaciones encapsuladas, en las que el robot lleva una población entera, o aproximaciones distribuidas en donde el robot solo lleva un código genético. Otra clasificación, más centrada en el tipo de paradigma evolutivo que se presenta en esta tesis, es la realizada por Eiben et al. [Eiben et al., 2010]. Esta clasificación no está basada en el concepto de evaluación dentro del robot, sino en el tiempo (cuándo se produce el proceso evolutivo, en tiempo de diseño o en tiempo de operación), espacio (dónde se produce el proceso evolutivo, dentro de los robots o fuera de los robots) y modo (cómo se realiza el proceso, de forma distribuida o de forma centralizada).

En esta tesis se propone una nueva clasificación de las aproximaciones evolutivas a la optimización de sistemas multi-robot teniendo en cuenta tres características básicas. La primera es si se permite la formación de equipos con controladores homogéneos o heterogéneos, haciendo referencia a la diferencia de controladores a nivel genético. Al tratar el problema de la optimización de sistemas multi-robot desde la perspectiva que se plantea en esta tesis, morfología y control por separado, esta es una característica fundamental ya que condiciona el tipo de sistema real al que se puede aplicar. En aproximaciones centradas en controladores homogéneos, se evoluciona un genotipo que posteriormente se clona en todos los robots del equipo, mientras que con controladores heterogéneos se evolucionan genotipos diferentes para cada robot. A nivel evolutivo, la principal ventaja de las aproximaciones homogéneas es que al evolucionar solo los parámetros para un individuo, se reduce la complejidad del espacio de búsqueda (se divide entre el número de robots el número de parámetros a optimizar). Es importante puntualizar que un genotipo idéntico no impide que los robots presenten comportamientos diferentes asumiendo roles diferentes aprovechando sus lecturas de sensores o basándose en un aprendizaje en su tiempo de vida. Otra posibilidad es que robots diferentes puedan expresar en su controlador partes diferentes de un genotipo idéntico. La heterogeneidad en cuanto a controladores permite comportamientos diferenciados en cada robot, pero a su vez incrementa el espacio de búsqueda, por aumentar el numero de parámetros a evolucionar. De entre estas dos alternativas, controladores homogéneos y heterogéneos, los equipos con controladores heterogéneos son a priori más adecuados cuando el problema a resolver es fácilmente divisible en sub-tareas ya que se permitiría que los controladores se especialicen. Se debe destacar que el algoritmo que soporta genotipos heterogéneos es más genérico y engloba al homogéneo, ya que

partiendo de controladores heterogéneos se puede converger hacia una solución que sea un controlador similar en cada uno de los robots, si es lo óptimo para la tarea. Finalmente, hay que destacar también que, como se comentó anteriormente, la morfología del sistema real de robots suele venir preestablecida, por tanto, impone el tipo de aproximación.

Además de esta característica básica, consideraremos si las aproximaciones realizan una evolución on-line u off-line, y por último, si es off-board u on-board. Las definiciones de estos tipos de evolución son las siguientes:

- Evolución on-line: el proceso de evaluación ocurre de forma continua y simultánea a los procesos evolutivos que se intercalan cada cierto tiempo en la evaluación.

- Evolución off-line: el proceso de evaluación es un proceso separado a los procesos evolutivos. Evaluación y evolución ocurren de forma secuencial: se evalúa y posteriormente se ejecutan los procesos evolutivos.

- Evolución on-board: un tipo de proceso evolutivo que se caracteriza porque todas las operaciones del algoritmo se ejecutan de forma independiente dentro de cada uno de los robots o dentro de cada proceso equivalente en simulación. Otro término para referirse a este tipo de evolución es "embodied" o descentralizada, ya que la evolución no depende de ningún elemento central como una evaluación externa de la calidad.

- Evolución off-board: un tipo de proceso evolutivo que se caracteriza porque existen operadores del algoritmo que se ejecutan externamente a los robots, por ejemplo, en una estación de trabajo conectada a los robots.

En la tabla 3.1 se muestra un resumen de los principales trabajos realizados organizados en base a estos criterios, pero excluyendo la categoría on-line y on-board, que se abordará más adelante en profundidad, debido a que es donde se enmarca el algoritmo presentado en esta tesis. La revisión de los principales trabajos en este campo se hará tomando como referencia esta tabla. La categoría principal de organización es si la técnica permite la obtención de controladores homogéneos o heterogéneos, y dentro de esta, se analizarán los sub-tipos definidos en la tabla.

Tabla 3.1: Aproximaciones basadas en evolución a sistemas multi-robot

Controladores heterogéneos			Controladores homogéneos
on-line		off-line	off-line
on-board	off-board	off-board	off-board
Silva (2012)	Stanley (2005)	Haynes (1995)	Luke (1998)
		Floreano (1998)	Quinn (2001, 2003)
		Uchibe (1998)	Potter (2001)
		Luke (1998)	Baldassarre (2003)
		Bongard (2000)	Dorigo (2004)
		Nelson (2004)	Mondada (2004)
		Eiben (2006)	Tuci (2006, 2008, 2014)
		Stanley (2008)	Ampatzis (2009)
		Agogino (2008)	Trianni (2011)
		Yong (2009)	Waibel (2009)
		Nitschke (2012)	
		Ferrauto (2013)	
		Gomes (2013)	
		Duarte (2016)	

3.2.2.1. Equipos con controladores homogéneos

En primer lugar, se describirán las aproximaciones que resuelven tareas con equipos con controladores homogéneos. Debido a que el controlador se debe clonar a todos los robots una vez evolucionado, y se evolucionan individuos secuencialmente, reiniciando el entorno, todas estas aproximaciones se clasifican como evolución off-line. Además, en todas ellas se realiza evolución off-board, ya que ésta se realiza externamente a los robots y solo se usan los robots reales o el simulador para evaluar los controladores. Los primeros ejemplos de equipos con controlador homogéneo se aplicaron a tareas de fútbol con robots. Luke et al. evolucionaron con programación genética un equipo para jugar al fútbol en la competición Robocup de robots virtuales [Luke et al., 1998b]. En este trabajo no se codifica por separado cada uno de los individuos sino que un genoma codifica al equipo completo, y de esta forma, se tiene una población de genomas que codifican todo el equipo. El controlador de cada robot estaba formado por dos árboles de programa, uno para chutar el balón y otro para moverse. Según el estado del jugador en el entorno se ejecuta un árbol u otro. Algunos comportamientos de bajo nivel están predefinidos como, por ejemplo,

ir a la pelota, chutar a portería e interceptar el balón. Quinn et al. proponen una tarea de movimiento coordinado realizada sobre dos robots Khepera virtuales para comparar el rendimiento de individuos evolucionados en equipo homogéneo o en equipo heterogéneo. A pesar de evolucionarse de forma heterogénea, la tarea se realiza siempre con un equipo de individuos clonados, pero se analiza cual de las dos formas de evolucionar obtiene mejor rendimiento. Se concluye que la evolución en equipos heterogéneos obtiene mejores resultados que la alternativa. Es decir, a pesar de ser un equipo homogéneo, los individuos evolucionados formando colaboraciones, realizan una división de roles cuando se forma un equipo clonado con ellos [Quinn, 2001]. Posteriormente los autores extendieron este trabajo a robots reales y presentaron uno de los primeros trabajos que realizaba un comportamiento coordinado y cooperativo de un sistema multi-robot real, también con un equipo homogéneo [Quinn et al., 2003]. En este caso la tarea de formación se realiza entre tres robots, tanto en simulación como en robots reales. Para evaluar un individuo se clona en los tres robots y se realizan varias evaluaciones en posiciones iniciales diferentes. El controlador de los robots es una red neuronal pulsada y se introducen dos tipos de operadores genéticos: un operador para modificar la estructura de la red (macro-mutación) y otro operador para mutar los valores de los parámetros (micro-mutación).

En [Baldassarre et al., 2003] se realiza una tarea de agregación y de ir hacia una luz con un equipo homogéneo de cuatro robots Khepera. Se evoluciona un genotipo que codifica los pesos de una red de neuronas artificiales y a la hora de evaluar, se clona este controlador en los cuatro robots. Se intenta realizar una simulación lo más real posible y para ello se realizan muestras de los sensores de los robots para utilizar en la simulación. Se observa en los resultados que los grupos de robots muestran diferentes estrategias para ir hacia la luz agrupándose de forma diferente y se observa especialización según la situación de cada robot en la formación.

Los equipos homogéneos han tenido el interés también de proyectos europeos de gran relevancia como el proyecto SWARM-BOT, que fue realizado entre 2001 y 2005. Para este proyecto se diseñó un robot denominado s-bot, preparado para acoplarse con otros y con gran interés en la auto-organización y la coordinación entre ellos. El s-bot cuenta con sensores preparados para su interacción local y un sistema de tracción dual con ruedas y tractor oruga para poder operar en superficies muy irregulares. También cuenta con un actuador para agarrar objetos o acoplarse a otros s-bot, de forma que su agregación es

capaz de moverse y empujar objetos pesados. Fue uno de los primeros sistemas en el que se agregaban más de dos módulos, ya que en trabajos previos la tendencia era realizar la agregación de dos módulos. Algunos de los experimentos fueron, por ejemplo, la realización de un movimiento coordinado de navegación con obstáculos. Cuatro robots s-bot forman una estructura de línea y cada uno de ellos cuenta con un sensor para saber cual es la diferencia entre su ángulo de tracción y el ángulo de tracción del grupo. Cada genotipo codifica todos las pesos de una red de neuronas artificiales y tiene una longitud de 80 bits, 8 bits por peso de la red. Se evoluciona en una población de 100 genotipos, en donde los 20 mejores genotipos generan 5 copias con un 3 % del genotipo aleatorio. Esta evolución se produce durante 100 generaciones [Dorigo et al., 2004]. En la figura 3.2 se muestra una foto de cuatro s-bot acoplados formando una línea.

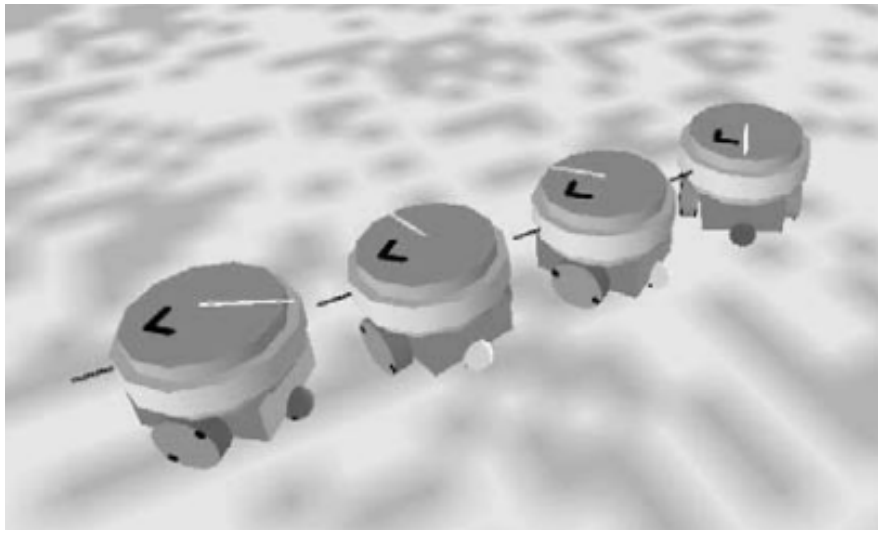


Figura 3.2: Cuatro s-bot acoplados formando una línea. Fuente: "Evolving Self-Organizing Behaviors for a Swarm-bot". [Dorigo et al., 2004]

Dentro del proyecto SWARM-BOT, también se ha desarrollado un simulador llamado *Swarmbot3d* para poder simular los robots s-bot con sus características particulares, como la capacidad de conectividad entre ellos, que es una característica que contemplada en otros simulador de robótica. Este simulador modela de forma realista todos los sensores de los robots s-bots y permite transferir los controladores directamente a los robots reales. Unos experimentos para compa-

rar el simulador desarrollado con el mundo real se puede encontrar en [Mondada et al., 2004], en donde los robots realizan una navegación conjunta por un entorno con una grieta y un experimento de subir un desnivel de diferente altura. Los conclusiones son que el simulador reduce el coste computacional de evaluar controladores mientras que un nivel tolerable de precisión en la simulación. Otro trabajo con los robots s-bot es el trabajo de Tuci et. al. en el que se resuelve una tarea que requiere coordinación para alcanzar a una presa y transportarla hasta una zona objetivo [Tuci et al., 2006]. Los controladores tienen dos submódulos: el módulo de ensamblaje, que permite acercarse y quedar agrupado a ciertos objetos, y el módulo de transporte que le permite moverse en la dirección deseada. El controlador del módulo de ensamblaje es una red de neuronas artificiales cuyos parámetros se evolucionan en una población y posteriormente se clona a cada uno de los s-bot que forma parte del experimento. En este trabajo se resuelven dos tipos de tarea: una tarea de transporte conjunto y una tarea de adaptación al entorno con la menor información a priori posible. De los mismos autores es el trabajo de coordinación de tres robots con hardware heterogéneo que deben seguir una luz evitando obstáculos [Tuci et al., 2008]. Es el primer trabajo en el que se obtenía un control homogéneo, es decir, un controlador clonado pero en robots con hardware heterogéneo. Los robots deben cooperar para realizar la tarea, ya solo uno de los tres robots tiene un sensor de luz pero no de infrarrojos (se encargaría de localizar la luz) mientras que los otros dos robots tienen sensores infrarrojos pero no de luz (se encargarían de la subtarea de evitar obstáculos). Los robots cuentan con altavoces y micrófonos para comunicarse a través del sonido para realizar la cooperación. Su controlador es una red de neuronas de tipo CTRNN (Continuous Time Recurrent Neural Network) que es un tipo de red muy utilizada en robótica evolutiva. Se codifican los parámetros de esta red en un genotipo de 84 valores reales y se evolucionan en una población de 80 genotipos. En la figura 3.3 se muestran los fotos usados, a la izquierda el primer tipo de robot con infrarrojos, a su derecha el segundo tipo de robots con sensor de luz y a la derecha el controlador de ambos.

Otro trabajo con controladores homogéneos es de Ampatzis et al., en donde se presenta una tarea de auto-ensamblaje entre dos robots con hardware idéntico pero que deben adoptar roles diferentes [Ampatzis et al., 2009]. Uno de los robots debe intentar acoplarse al otro y el otro robot debe facilitar este acople. Para ello deben coincidir en una orientación en la que es posible este acoplamiento. El controlador es una red CTRNN y cada genotipo codifica todos los parámetros de esta red con 263 valores reales. Estos genotipos se evolucionan en una población

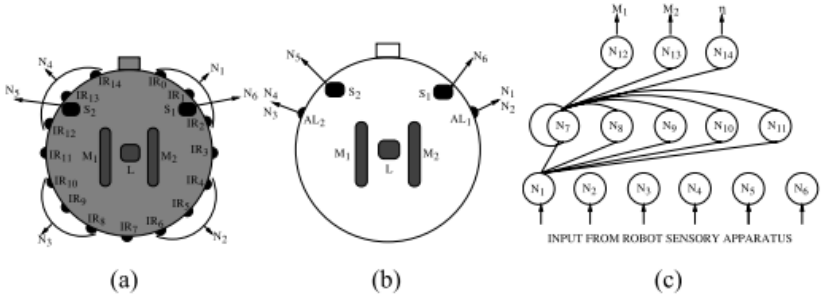


Figura 3.3: Robots usados en los trabajos de Tuci. "Evolving Homogeneous Neurocontrollers for a Group of Heterogeneous Robots: Coordinated Motion, Cooperation, and Acoustic Communication" [Tuci et al., 2008]

de 100 genotipos durante 10000 generaciones en un simulador para acelerar el proceso de evolución. Una vez finalizada esta evolución, se transfieren los controladores a los robots reales y se comprueba que los resultados concuerdan con los obtenidos en simulación. En la figura 3.4 se muestra una fotografía de los robots reales usados. Un último ejemplo de gran relevancia con equipos homogéneos es el trabajo de Trianni y Nolfi, en el que se estudia la sincronización de tres robots con la misma morfología. Su controlador es una red de neuronas artificiales y se evoluciona un controlador en una población de 100 genotipos que posteriormente se clona a los tres robots [Trianni and Nolfi, 2011].

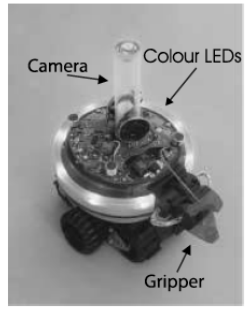


Figura 3.4: Fotografía de los robots s-bot usados por Ampatzis et al. Fuente: "Evolving Self-Assembly in Autonomous Homogeneous Robots: Experiments with Two Physical Robots". [Ampatzis et al., 2009]

Como se ha visto en todos estos ejemplos, evolucionar equipos homogéneos ha proporcionado resultados satisfactorios en tareas con sistemas multi-robot, principalmente porque es más sencillo que evolucionar equipos heterogéneos, como veremos a continuación.

3.2.2.2. Equipos con controladores heterogéneos

A continuación se describirán trabajos relevantes que evolucionan equipos con controladores heterogéneos, empezando por trabajos con evolución off-line, es decir, que realizan evaluaciones reiniciando las condiciones del entorno mientras se intercalan los operadores genéticos. Estas aproximaciones también son off-board, ya que en ellas la evolución se produce externamente a los robots.

Uno de los primeros trabajos evolucionando equipos con controladores heterogéneos es el de Haynes et al. [Haynes et al., 1995] en el que un equipo de cuatro depredadores intenta cazar a una presa en un entorno simulado dividido en celdas. La presa intenta escapar siempre del depredador más cercano y solo se produce la captura si todos los depredadores están en celdas adyacentes a la presa. Se utilizan el paradigma de STGP, que es una extensión de la programación genética clásica, para evolucionar los programas de cada uno de los depredadores. El método de codificación es un cromosoma que contiene k programas para todo el equipo. En este trabajo también se comparan diferentes formas de realizar los operadores de recombinación, por ejemplo, sustituyendo porciones de los programas, o sustituyendo programas enteros procedentes de otros equipos. Los resultados muestran estrategias que obtienen mejor rendimiento que cualquier solución manual, y respecto a las diferentes formas de recombinación, se concluye que es beneficioso que todos los programas del equipo participen en las recombinaciones para que el aprendizaje sea mas rápido.

Uno de los primeros trabajos con robots reales en que se evolucionan controladores heterogéneos es el de Floreano et at. [Floreano and Nolfi, 1997]. En este trabajo se presenta también una tarea de depredador presa con dos robots Khepera, de forma que se evolucionan sus dos correspondientes poblaciones de forma competitiva. En la figura 3.5 se muestran los robots Khepera usados en este experimento. El depredador tiene sensores más avanzados que la presa, como visión, mientras que la presa tiene sensores más simples pero se puede mover mucho más rápido que el depredador. Se estudia el efecto denominado “efecto de la reina roja”, que se produce cuando dos poblaciones se modifican mutuamen-

te el espacio de búsqueda ya que compiten contra la otra población. El control de los robots es una red de neuronas artificiales y se codifican en 160 bits. Se evolucionan dos poblaciones de 100 individuos durante 100 generaciones. Cada individuo se evalúa en el entorno contra el mejor de las últimas 10 generaciones de los competidores. Cada vez que se realiza una evaluación, el depredador y la presa se sitúan sobre una linea horizontal a una distancia fija pero con orientaciones aleatorias. Se concluye que se pueden evolucionar comportamientos en los robots como seguimiento, evitar obstáculos, reconocimiento visual, de forma espontánea y que la coevolución competitiva es un aspecto de gran interés en computación evolutiva. Se puede definir coevolución se produce cuando la calidad de un individuo depende de su interacción con otros individuos, ya sea cooperativa (objetivo común) o competitiva (objetivos diferentes).

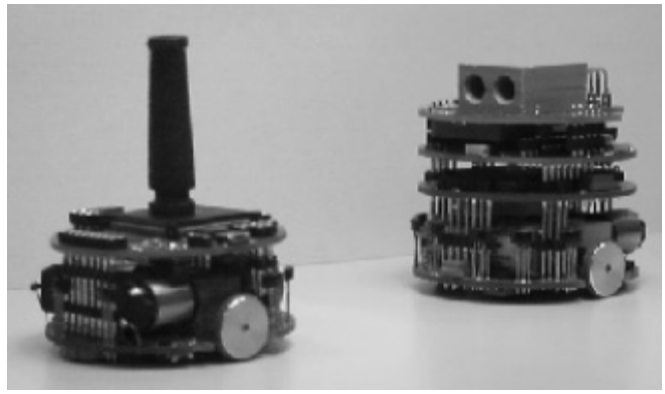


Figura 3.5: Fotografía de los robots Khepera usados por Floreano et al. Fuente: "God Save the Red Queen! Competition in Co-Evolutionary Robotics" [Floreano and Nolfi, 1997]

Otro trabajo destacable con equipos genéticamente heterogéneos es el de Luke et. al [Luke et al., 1998a], que propone enfrentar dos equipos de robots en un entorno de fútbol simulado. Se parten de comportamientos predefinidos y se evolucionan con programación genética. El genotipo codifica a todo el equipo y los equipos se evolucionan competitiva contra otros equipos de la población. Con el fin de disminuir el número de evaluaciones y no evaluar cada equipo contra todos, se forman pares de equipos al azar para evaluar. De esta forma se evalúan dos equipos a la vez y en una población de n equipos se necesitan $n/2$ evaluaciones. La calidad obtenida por un equipo está basada en

la diferencia de goles entre los dos equipos. Los resultados obtenidos son que los equipos evolucionados aprenden la tarea de forma satisfactoria y que son capaces de ganar a equipos realizados a mano. Otro ejemplo también en el dominio de fútbol con robots es el de Uchibe et al. [Uchibe et al., 1998], en el que dos robots cooperan para marcar un gol, y otro robot es el defensor y su objetivo es impedir el gol. Al igual que en el trabajo de Luke et al., el método empleado es programación genética, y como en otras aproximaciones basadas en programación genética, se predefinen a mano unos comportamientos básicos: tirar a gol, pasar, esquivar obstáculo, buscar la pelota. Cada robot se evoluciona en una población aislada y para evaluar conjuntamente se seleccionan individuos de cada una de las poblaciones. En la función de calidad se introducen términos para valorar las veces que se ha golpeado la pelota un robot, de forma que no se recompensan comportamientos pasivos.

También se emplea programación genética en el sistema Legion [Bongard, 2000]. Igual que en la aproximación de Luke et al., se codifica todo el equipo conjuntamente, pero en este caso con dos o más expresiones *s*. Las expresiones s son una notación para representar una estructura de árbol. La primera expresión s realiza particiones en el grupo de agentes para dividirlos en clases de comportamientos y las siguientes expresiones s son las que corresponden a los comportamientos de cada partición. Se introduce una medida de heterogeneidad de los equipos basada en la calidad y se analiza el algoritmo en dos tareas: una tarea de optimización de rutas de reparto de correo para minimizar el correo no atendido y una tarea de búsqueda de comida por una colonia virtual de hormigas. En la primera tarea, el sistema tiende más a la heterogeneidad del equipo de agentes que para la segunda tarea y se concluye que la heterogeneidad del equipo es una propiedad que depende del dominio.

En [Nelson et al., 2004], Nelson et al. proponen una versión de un juego de capturar la bandera que enfrenta a dos equipos de robots. Un equipo de robots debe defender su meta pero simultáneamente debe atacar la del equipo contrario. El entono es un laberinto reconfigurable y los robots tienen una cámara para poder esquivar obstáculos y distinguir robots del mismo equipo o del otro según el color. El controlador es una red de neuronas artificial y la forma de evaluar un controlador en el entorno es enfrentándolo contra otro individuo de la misma población, es decir, competitivamente dentro de la población. Cada par de controladores se enfrentan dos veces entre sí. Una vez finalizados los enfrentamientos, se seleccionan los controladores que mejor resultado han obtenido para

aplicarles los operadores genéticos y producir la siguiente generación. El interés de este trabajo era aplicar la aproximación evolutiva en una tarea compleja, como es un juego competitivo, y los resultados demuestran que los equipos obtenidos en simulación ofrecen un comportamiento similar al transferirse a los robots reales.

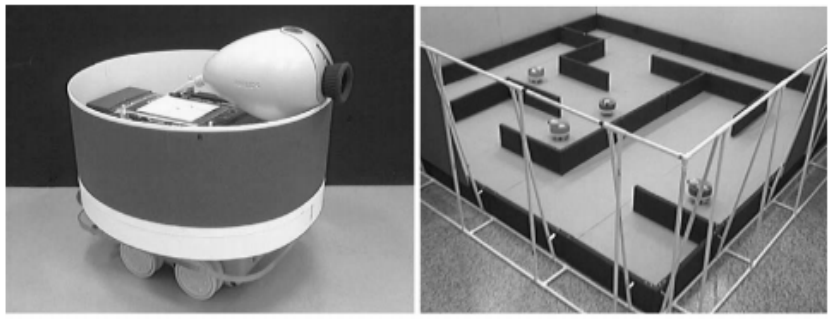


Figura 3.6: Fotografía de los robots usados por Nelson et al. Fuente: "Evolution of neural controllers for competitive game playing with teams of mobile robots" [Nelson et al., 2004]

Existen diversos trabajos que estudian las condiciones para que los equipos heterogéneos creen robots especialistas, como por ejemplo el trabajo de Potter et al. [Potter et al., 2001]. En este caso, se concluye que no es aumentar la dificultad de la tarea lo que produce que se creen especialistas, sino el número de capacidades requeridas para realizarla. En concreto, la tarea que se presenta consiste en un grupo de robots que hacen de pastores y tienen que forzar a otro robot, con el rol de oveja, a que entre en un corral. El robot con el rol de oveja tiene un comportamiento fijo de intentar escaparse a través de uno de los lados del entorno, evitando los obstáculos y a los robots pastores. Se compara evolucionar controladores genéticamente homogéneos o heterogéneos y los resultados muestran que evolucionar equipos heterogéneos puede ser beneficioso pero tiene mayor coste computacional.

En este ámbito, una comparativa entre equipos genéticamente homogéneos y equipos heterogéneos se puede encontrar en [Waibel et al., 2009]. Aquí se consideran cuatro formas de evolucionar un sistema multi-robot para una tarea cooperativa. Se parte una población que está formada por un conjunto de equipos. Las cuatro formas de crear un equipo a partir de este conjunto de

equipos son: equipos homogéneos con selección individual, equipos homogéneos con selección a nivel de equipo, equipos heterogéneos con selección individual, y equipos heterogéneos con selección a nivel de equipo. El experimento que se propone para comparar estas alternativas es una tarea de búsqueda de comida con 10 robots. La tarea consistía en transportar dos tipos de comida a una zona marcada. En el entorno se sitúan dos tipos de comida: un tipo de comida grande que necesitaba ser transportada por dos robots cooperando y un tipo de comida pequeña que podía transportar un robot individualmente. Se realizan varios experimentos: una tarea de búsqueda de comida solo con el tipo de comida pequeña, una tarea de búsqueda de comida cooperativa con el tipo de comida grande y una tarea de búsqueda de comida de forma altruista. Las conclusiones fueron que los equipos heterogéneos con selección individual obtenían mejor rendimiento en las tareas que no se requería cooperación, y los equipos homogéneos obtuvieron mejor rendimiento en la tarea con cooperación, siendo la mejor opción en tareas cooperativas que no requieran especialización. Otra comparativa similar a esta es la de Tuci y Trianni [Tuci and Trianni, 2014], en la que la tarea, a diferencia de la comparativa de Waibel et al., requiere especialización. En este trabajo se buscan los controladores para un equipo homogéneo de robots y se comparan las dos formas de evolucionar: con equipos homogéneos o con equipos heterogéneos. La tarea es una tarea inspirada por las tareas de los insectos sociales en la que dos robots Khepera simulados deben realizar simultáneamente vigilancia del nido y búsqueda de comida. El controlador de los robots es una red CTRNN codificada en un vector de 135 valores reales. La principal conclusión de este trabajo es que evolucionar en equipos homogéneos obtiene mejor rendimiento que evolucionar en equipos heterogéneos, lo cual difiere de las conclusiones obtenidas en la comparativa hecha por Quinn et al. una década antes.

Uno de los problemas a la hora de evolucionar controladores para un sistema multi-robot heterogéneo, y que ha sido tratado en diversas ocaciones, es el problema de asignar la calidad de cada miembro del equipo. Si se mide dicha calidad a través de una medida global, utilizar esta medida global para cada individuo supone evaluar de forma incorrecta a algunos individuos. En algunos casos, la evaluación puede ser favorable al individuo (con un equipo de controladores casi óptimos) y en otros casos puede ser desfavorable (individuo óptimo en un equipo de mal rendimiento). Para solucionar esto, una de las aproximaciones es realizar múltiples evaluaciones de un mismo genotipo formando equipos diferentes, teniendo la desventaja de aumentar mucho el tiempo necesario para

evaluar a toda la población. Teniendo múltiples evaluaciones de un individuo se puede realizar una media de todas las evaluaciones, usar la mejor evaluación o la peor. Otra de las aproximaciones, que también requiere más de una evaluación, es calcular la aportación individual al rendimiento de un equipo como la diferencia entre una evaluación con el individuo y sin él. Esta última tiene la ventaja de que solo hay que realizar dos evaluaciones y que la estimación de la aportación puede ser más fiable que varias evaluaciones [Wolpert and Tumer, 2001]. Un trabajo que aplica esta evaluación de calidad en la evolución de un equipo de controladores heterogéneos de un sistema multi-robot, es el realizado por Agogino et al. [Agogino and Tumer, 2008]. En este caso se describen los diferentes esquemas de evolucionar un sistema multi-robot que para los autores son:

1. de entre una población de controladores, uno de ellos controla a todos los robots
2. de entre una población de controladores, cada controlador puede controlar [0,n] robots (algunos robots comparten controlador)
3. cada robot tiene su población de controladores

Se propone una tarea de coordinación que es una búsqueda de puntos de interés por un equipo de robots. Para hacer la tarea más compleja, el entorno es dinámico y los puntos de interés se mueven continuamente, los sensores de los robots son ruidosos y la comunicación entre robots es limitada. Cada robot tiene una población de 10 genotipos que codifican un perceptrón multicapa, un tipo de red de neuronas artificiales. Los pasos del algoritmo son los siguientes:

1. para cada robot, se selecciona para evaluar uno de los controladores de su población con una estrategia ϵ-greedy
2. se realiza una ligera mutación sobre este controlador, se evalúa por un período de tiempo fijo y se devuelve a la población
3. se elimina el peor controlador de la población de ese robot

Se analiza el algoritmo en diferentes experimentos: sin ruido en los sensores, con ruido en los sensores, con menos capacidad de comunicación entre robots y con diferentes funciones de evaluación (una de rendimiento global, una de rendimiento global teniendo en cuenta al resto de robots, y una diferencial calculada

como en el trabajo de Wolpert et al.). Los resultados obtenidos muestran que la función de evaluación diferencial es adecuada para evolucionar la coordinación de sistemas multi-robot en dominios muy complejos.

Dentro de la categoría de algoritmos de evolución off-line y off-board para obtener controladores heterogéneos hay que destacar la familia de los algoritmos CCEA (Cooperative Coevolution Evolutionary Algorithm). Estos algoritmos utilizan la coevolución como herramienta para abordar problemas complejos, como problemas de alta dimensionalidad. Los algoritmos CCEA han sido aplicados originalmente a problemas de optimización generales pero su arquitectura multi-componente supone que sean la base de otros algoritmos para la evolución de controladores en sistemas multi-robot. Un ejemplo es la arquitectura de sub-componentes de Potter [Potter and De Jong, 2000]. Esta arquitectura está basada en diferentes sub-componentes (denominados especies) que son poblaciones ejecutando un algoritmo evolutivo cada una. Esta estructura hace que sea directamente aplicable a sistemas multi-robot asociando una población a cada uno de los robots, como se puede ver en trabajos posteriores de los mismos autores [Potter et al., 2001]. Una de las aportaciones de esta arquitectura es que adapta en tiempo de evolución su número de especies, ya que puede crearlas y destruirlas según si corresponde. Se crean nuevas especies si la evolución no mejora en una ventana de tiempo y se destruyen cuando se comprueba que las aportaciones de los individuos que pertenecen a ella son nulas. De esta forma soluciona el problema de establecer a priori el número de especies. A pesar de evolucionar cada componente de forma aislada como si fueran islas, la evaluación de un individuo depende de formar colaboraciones con los otros componentes. No existe el problema de la asignación de calidad ya que cada individuo de una población se evalúa con los mismos individuos de los otros componentes.

Como se ha podido comprobar, en muchos de los trabajos aquí descritos, los controladores de los robots son redes de neuronas artificiales que se evolucionan (neuroevolución), pero con topología fija, es decir, se evolucionan los parámetros de la red pero manteniendo fija la morfología de la red y conexiones. Un algoritmo que solventa esta limitación es el algoritmo de neuroevolución NEAT [Stanley and Miikkulainen, 2002], porque permite evolucionar a la vez la morfología de las redes y sus parámetros, y por este motivo es uno de los algoritmos más relevantes de neuroevolución en la literatura. Debido a que a que se evoluciona la morfología de cada uno de los controladores, los controladores resultantes son heterogéneos, no solo en parámetros sino tambien en

morfología. En este algoritmo NEAT, las estructuras de las redes de neuronas que evolucionan son inicialmente simples pero los operadores genéticos producen modificaciones estructurales que convierten las estructuras de red en más complejas si el problema así lo requiere. El algoritmo NEAT es originalmente un algoritmo de evolución off-line, pero ha dado lugar a muchos otros algoritmos derivados, algunos de ellos aplicados a sistemas multi-agente y a sistemas multi-robot, como el algoritmo rtNEAT, que es una versión de NEAT para evolución on-line. El algoritmo rtNEAT fue presentado por Stanley y Mikkulainen en la evolución de controladores de un equipo de robots virtuales para jugar dentro de un videojuego [Stanley et al., 2005]. En este trabajo, el equipo de robots virtuales evolucionado por rtNEAT se enfrenta al equipo de robots que maneja el jugador. A diferencia del algoritmo NEAT, en el que en cada generación se modifica toda la población, en el algoritmo rtNEAT se generan individuos nuevos en un intervalo de tiempo fijo y se reemplazan los peores individuos con los nuevos. Una desventaja del algoritmo NEAT, también presente en rtNEAT, es que están basados en información centralizada, por ejemplo, marcadores históricos, y medidas de diferencias entre genotipos.

Un algoritmo derivado del algoritmo NEAT y que ha sido aplicado a un enjambre de robots, es el algoritmo Progressive Minimal Criteria Novelty Search (PMCNS) [Gomes et al., 2013]. Este algoritmo combina criterios de calidad y criterios de novedad de las soluciones y está basado en el método denominado Minimal Criteria Novelty Search (MCNS) [Lehman and Stanley, 2010]. El método MCNS es una modificación del algoritmo NEAT en el que los individuos deben cumplir un criterio mínimo para ser aptos para reproducirse. La inspiración del método MCNS es la técnica de Novelty Search [Lehman and Stanley, 2008], que se basa en guiar la evolución no por la calidad de las soluciones, sino por su novedad con respecto las soluciones existentes. En el caso del experimento presentado en el trabajo original del método MCNS, que era un experimento de navegación de sólo robot en un laberinto, el criterio mínimo es que el robot esté dentro del laberinto una vez evaluado el controlador, que es lo que cumplen los comportamientos que son capaces de navegar el laberinto. De esta forma, todos los comportamientos que no cumplen este criterio no son aptos para reproducirse. En el trabajo de Gomes et al., se combinan los criterios de novedad y calidad para producir los controladores de un equipo de robots en varias tareas: una de agregación y una de compartir recursos, en la que los robots deben acceder a una zona de recarga en la que solo puede estar un robot recargándose y se pueden producir conflictos a la hora de acceder a este

recurso. En la tarea de agregación los métodos basados en calidad y en Novelty Search obtienen rendimientos similares, pero en la tarea de compartir recursos, los métodos basados en Novelty Search obtienen mejor rendimiento ya que los métodos basados en calidad se quedan atrapados en un máximo local.

Otro ejemplo de aplicación del algoritmo NEAT a un sistema multi-robot es el trabajo de Duarte et al., centrado en un enjambre de robots acuáticos de superficie [Duarte et al., 2016]. En este trabajo se proponen varias tareas que son habituales en sistemas colectivos: una tarea de ir a un punto de interés evitando obstáculos, una tarea de dispersión, una tarea de agregación y una tarea de vigilancia. Los controladores de los robots se evolucionan en simulación y posteriormente se prueban en el entorno real. Los controladores obtienen un comportamiento y rendimiento en la tarea similar que en simulación y queda demostrado que se pueden evolucionar comportamientos para un sistema multi-robot en un entorno ruidoso sin condiciones controladas de un laboratorio.

Otro método de neuroevolución es el algoritmo Multi-Agent ESP presentado por Yong y Mikkulainen [Yong and Miikkulainen, 2001], que es una extensión del algoritmo ESP de Gomez y Mikkulainen para aplicarse en un sistema multi-agente. El algoritmo ESP es un algoritmo de neuroevolución que evoluciona poblaciones de neuronas para formar una única red de neuronas artificial. El algoritmo Multi-Agent ESP se ha estudiado en el dominio depredador presa, en particular en una tarea en la que tres depredadores deben cooperar para cazar a la presa, que escapa del depredador más cercano. En este ejemplo se prueban dos aproximaciones, una aproximación en los depredadores tienen el mismo controlador y otra en la que tienen controladores diferentes, de forma que se requiere evolucionar tres controladores. Cada uno de estos controladores se forma siguiendo el algoritmo ESP. Se comprueba que la aproximación con tres controladores es más eficiente y robusta que evolucionar un solo controlador. El Multi-Agent ESP, a pesar de ser un método que solo se ha aplicado para evolucionar en simulación, podrían ser aplicado a un sistema multi-robot real si se transfieren los controladores obtenidos.

Algunos trabajos más recientes de neuroevolución son el algoritmo odNEAT [Silva et al., 2012] [Silva et al., 2015], una versión distribuida del algoritmo NEAT para ser aplicada a evolución on-line de sistemas multi-robot. Al igual que el algoritmo rtNEAT, el algoritmo odNEAT evoluciona de forma on-line, pero con la novedad de sin ningún tipo de elemento central. En odNEAT, los robots tienen un nivel de energía virtual que está asociado a su comportamiento en la

tarea y su nivel de calidad es una media de una muestra de valores de energía en un intervalo de tiempo. Cada robot tiene un repositorio de cromosomas y cuando al cromosoma activo se le acaba la energía, un cromosoma del repositorio pasa a ser el activo. El algoritmo fue analizado en una tarea de agregación con robots e-puck simulados y se comparó contra el algoritmo rtNEAT. Se comprobó que obtiene resultados similares a este algoritmo con la ventaja de utilizar únicamente información local y la no dependencia de un punto central.

Otro algoritmo basado en neuroevolución y que se ha aplicado a sistemas multi-robot heterogéneos es el algoritmo CONE (Collective NeuroEvolution) [Nitschke et al., 2010]. Las principales contribuciones del algoritmo CONE es que resuelve las tareas a través de especialización emergente, que es un tema de relevancia en esta tesis, y que introduce dos métricas para este fin. Una de ellas es una métrica de diferencia de genotipos, que tiene en cuenta si la diferencia entre dos genotipos es menor que un umbral. Otra métrica es una métrica de especialización que permite decidir si dos controladores de poblaciones diferentes son suficientemente parecidos en comportamiento para recombinarse. El funcionamiento del algoritmo CONE es el siguiente:

1. se parte de una población de genotipos, cada uno conteniendo una subpoblación de neuronas

2. se seleccionan neuronas de diferentes genotipos y se forma tantos controladores como robots en el sistema, de forma que cada controlador es diferente al resto

3. se evalúan estos controladores en la tarea un periodo de tiempo fijo. Cada neurona de diferentes genotipos que participa en este controlador obtiene un valor de calidad que es el promedio de la calidad obtenida por el controlador. Se evalúan todas las neuronas de cada genotipo al menos una vez

4. se aplican los operadores genéticos a nivel de subpoblación de neuronas

La principal diferencia de este método con otros de neuroevolución es que los operadores genéticos se aplican a nivel de subpoblación de neuronas, de forma que se seleccionan las mejores neuronas de cada subpoblación para producir los nuevos genotipos. Es decir, los operadores genéticos no se aplican sobre los controladores sino sobre las neuronas. Este algoritmo se ha aplicado a diversas

tareas con robots simulados. Uno de los primeros trabajos con este algoritmo es una tarea de descubrimiento de rocas rojas dispersas por el entorno [Eiben et al., 2006]. Cuando un robot descubre una roca roja, queda marcada como descubierta. Se compara el algoritmo con un algoritmo de neuroevolución tradicional, y el algoritmo CONE obtiene mejores resultados.

Una tarea más compleja en la que se ha aplicado el algoritmo CONE es una tarea de construcción colectiva con robots Khepera simulados [Nitschke et al., 2012]. En esta tarea los robots deben cooperar para construir una estructura a partir de unos bloques, en una determinada secuencia. Existen dos tipos de bloque, de tipo A y de tipo B, distinguibles por una luz y los robots deben mover estos bloques hacia una zona de construcción en una determinada secuencia fija. Si el bloque no es el que corresponde en esa secuencia, no es posible colocarlo en la zona de construcción. La calidad del equipo de robots es el número de bloques en la zona de construcción después del tiempo de vida de un equipo. Los controladores se etiquetan como especializados si la mayor parte de su tiempo realizan la misma acción (mismas salidas de motores). Se comprueba que existen robots especializados como constructores capaces de recoger bloques fuera de secuencia colocados por robots sin especializar y colocarlos en la secuencia correcta. Se compara el algoritmo CONE contra otros algoritmos (Multi-Agent ESP y CCEA) y éste obtiene mejores resultados en esta tarea de construcción colectiva. Igual que en otros ejemplo del algoritmo CONE, se concluye que el algoritmo es adecuado para evolucionar comportamientos colectivos especializados incluso cuando no se sabe que tipo de especialización es la beneficiosa.

Otro estudio que también aborda la especialización es el de Ferrauto et al. [Ferrauto et al., 2013], en el que se comparan cuatro algoritmos evolutivos en dos tareas cooperativas con dos robots Khepera simulados. Las dos tareas son una tarea de aproximarse a la luz y una tarea de movimiento coordinado. La tarea de movimiento coordinado puede ser resuelta de forma más eficiente con especialización mientras que la tarea de aproximación a la luz se resuelve de forma más eficiente asignando y cambiando roles de forma dinámica. Los cuatros algoritmos que se comparan son: un algoritmo con grupo homogéneo, en donde se evoluciona un controlador y se clona, un algoritmo con grupo heterogéneo, en el que se codifican los dos controladores en un mismo genotipo, un algoritmo con múltiples poblaciones, cada una de ellas correspondiendo a un robot, y un algoritmo con una sola población muy numerosa, en el que los genotipos codifican a un solo robot. Para los últimos dos algoritmos menciona-

dos se analizan dos configuraciones: grupos fijos o grupos temporales. El único de los cuatro algoritmos que consigue obtener especialización e intercambio de roles cuando corresponde en cada tarea, es el algoritmo con una sola población y grupos temporales.

Como conclusión a este sub-apartado, se debe destacar la gran relevancia que ha adquirido en los últimos años la evolución de sistemas multi-robot heterogéneos, y la complejidad que se ha logrado alcanzar en las tareas a resolver. Son destacables los algoritmos basados en coevolución cooperativa, ya que obtienen resultados muy satisfactorios con un enfoque totalmente emergente y heterogéneo. Hasta el momento no se ha tratado aquí la optimización de la morfología de los robots, que en todos los casos anteriores venía predefinida. Este es el objetivo del siguiente sub-apartado.

3.2.2.3. Optimización de la morfología

Los algoritmos evolutivos han sido las principales aproximaciones aplicadas en la optimización de la morfología, tanto en la del controlador como hemos visto con el caso de la neuroevolución, como en la de la morfología física, principalmente a través de robots modulares. Un ejemplo de evolución de morfología y control es el trabajo de Faiña et al. [Faíña et al., 2013], que se basa en un algoritmo evolutivo propio que evoluciona las ramas, pesos y nodos en una representación de árbol. En lugar de utilizar un conjunto de módulos hardware homogéneos, este sistema está formado por cuatro tipos de módulos heterogéneos: dos de ellos producen movimientos lineales y los otros dos producen rotatorios. Se analiza el sistema en dos experimentos: uno de transportar una carga en diferentes tipos de superficies (plana, irregular) y otro de pintado de una pared vertical cubriendo el mayor área posible. Los resultados obtenidos demuestran que la aproximación empleada es factible y obtiene soluciones realistas. Este trabajo muestra cómo la optimización de la morfología permite un control más simple para llevar a cabo una tarea real.

Otro ejemplo son los organismos multi-robot del proyecto Symbrion. El organismo multi-robot está formado por módulos homogéneos que pueden agregarse y desagregarse según los requerimientos de la tarea. Los controladores funcionan tanto a nivel individual como agregado [Levi and Kernbach, 2010]. Un trabajo destacado con este tipo de organismo multi-modular es el Hamann et al., en el que se evoluciona el sistema para que sea capaz de desplazarse a partir

de auto-configurar sus módulos y controlarlos tanto a nivel individual como en agrupaciones [Hamann et al., 2010].

Como conclusión a este apartado de antecedentes, se debe destacar que la aplicabilidad a sistemas multi-robot reales de muchos de los algoritmos evolutivos revisados es limitada, y prueba de ello es que la mayoría están aplicados únicamente en simulación, y lo que es más patente, partiendo del diseño de un algoritmo y aplicándolo a un sistema multi-robot adaptado, y no al revés. Aunque la potencialidad de las aproximaciones emergentes es claramente mayor que la de las intencionales a la hora de tratar con problemas reales y generales, aún queda mucho trabajo por delante para lograr que la optimización de sistemas multi-robot mediante algoritmos evolutivos sea realmente aplicable. En esta tesis, con el afán de dar pasos en este sentido, se busca por encima de todo que el algoritmo a desarrollar pueda ser aplicado a un sistema multi-robot lo más realista posible, de modo que el propio algoritmo no limite el tipo de aplicación. Con este objetivo, en el siguiente capítulo, se describirán en más detalle las especificaciones que consideramos que debe cumplir un algoritmo para ser aplicado a la optimización de un sistema multi-robot real y general, y se realizará una revisión bibliográfica de las aproximaciones evolutivas más prometedoras en este sentido, aquellas basadas en el paradigma de Embodied Evolution.

Capítulo 4

Requisitos del algoritmo de optimización

En este capítulo se establecerán las especificaciones de diseño para el algoritmo a desarrollar. Dichas especificaciones parten de las características de los sistemas multi-robot reales sobre los que se aplicará el algoritmo, de modo que permita optimizar un sistema general de este tipo.

4.1. Especificaciones y requisitos de diseño

La mayor parte de los algoritmos evolutivos utilizados en la optimización de sistemas multi-robot que fueron revisados en el apartado anterior, parten de un análisis poco profundo del tipo de sistema real sobre el que serán aplicados. Se asume un sistema multi-robot genérico, formado por varios robots con capacidades estándar de sensorización y actuación, y se comienza el desarrollo del algoritmo. En dicho proceso, el algoritmo debe ser adaptado y mejorado en diversos aspectos, lo cual implica, en multitud de ocasiones, una redefinición del sistema multi-robot real, normalmente relajando alguna especificación del mismo. Esto conlleva que el grado de aplicabilidad final del algoritmo sea muy limitado, ya que funciona en condiciones poco transferibles a la realidad. La principal evidencia de este problema está en el reducido número de artículos de los revisados en el apartado anterior que realmente han dado lugar a un sistema

multi-robot en explotación. Ejemplos típicos de qué condiciones o especificaciones se relajan serían: uso de equipos homogéneos, aprendizaje en tiempo real, comunicaciones y operaciones centralizadas, entornos estáticos, etc.

En lugar de seguir este tipo de aproximación de abajo a arriba, partiendo del desarrollo de un algoritmo que posteriormente se aplica en un sistema multi-robot, y por tanto, condicionando dicho sistema multi-robot, en esta tesis se enfoca el problema de arriba a abajo. Partiremos, pues, de una caja negra que representa el sistema multi-robot, pretendiendo que éste sea lo más realista posible en cuanto a su realización práctica. En este sentido, tras haber analizado en profundidad la bibliografía y los casos de éxito en sistemas multi-robot, podemos establecer las características básicas que debe tener un sistema de este tipo para ser transferible a la práctica:

1. Teniendo en cuenta criterios prácticos, como coste económico y mantenimiento reducido, los sistemas multi-robot deben ser simples, a nivel individual, tanto en hardware como en software. Es más realista considerar un sistema multi-robot compuesto por robots simples dedicados a tareas simples, que alcanzan su complejidad al combinar sus comportamientos, que robots complejos capaces de realizar varias tareas difíciles, de modo que si uno falla o se estropea, su arreglo o restitución se vuelve muy costoso. Esto no quiere decir que los robots deban ser homogéneos, de hecho, no se establecerá ninguna limitación en este sentido.

2. Debe estar compuesto por robots con alto grado de autonomía, porque esto beneficia a la fiabilidad y escalabilidad del sistema, y reduce los problemas relativos a la comunicación. Es decir, aunque los sistemas centralizados son más simples de implantar inicialmente, su aplicabilidad práctica es más reducida.

3. Debe tener un sistema de control adaptativo, de modo que pueda reaccionar a los cambios que se produzcan en el entorno o en la tarea en tiempo real. Un sistema multi-robot diseñado para entornos estáticos o que deba ser rediseñado en caso de que se produzcan cambios no es práctico.

Partiendo de estas especificaciones, se pueden establecer los requisitos de diseño para el algoritmo evolutivo que permita lograrlas:

1. Debe permitir obtener controladores heterogéneos, porque son más flexibles y permiten reducir la complejidad de los individuos ya que pueden

especializarse en sub-problemas del problema a resolver. Además, la obtención de controladores heterogéneos permite utilizar sistemas multi-robot con morfología homogénea o no, de tal forma que es una aproximación más general.

2. Debe realizar una evolución on-board, ya que las aproximaciones off-board dependen de información centralizada lo cual limita la autonomía individual de los robots.

3. Debe permitir evolución on-line, ya que el tiempo de respuesta para adaptarse a cambios en el entorno es fundamental.

En resumen, el algoritmo evolutivo para optimizar el tipo sistema multi-robot objetivo debe realizar evolución on-line y on-board, así como permitir la obtención de soluciones heterogéneas. En este sentido, nos referimos a soluciones heterogéneas tanto a nivel de control como a nivel de morfología. En una primera aproximación, *no se tratará en esta tesis el diseño morfológico del sistema multi-robot*, sino que nos centraremos en el estudio de la coordinación. Debemos destacar que dicho diseño morfológico ha sido parcialmente estudiado en el pasado en un trabajo donde se obtiene la composición óptima del equipo a partir de varios tipos de robot predefinidos, así como el número de robots de cada tipo para resolver la tarea de forma óptima [Trueba et al., 2011]. También se ha realizado la optimización de la morfología en un un problema de optimización de fletes en el que se evoluciona tanto los parámetros físicos como el número de vehículos necesarios para resolver la tarea en diferentes condiciones de diseño [Prieto et al., 2016].

Analizando los requisitos anteriores desde una perspectiva más simple, el algoritmo a desarrollar debe permitir que los robots que forman el equipo sean lo más generales y autónomos posible, y también debe soportar el proceso de optimización en tiempo de ejecución. Con estas mismas premisas surgió a finales de los años 90 el paradigma evolutivo denominado Embodied Evolution, que será es que se utilizará en esta tesis, propuesto por Watson et al. [Watson et al., 1999]. Embodied Evolution es un paradigma de computación evolutiva inspirado en la evolución natural, donde los individuos que forman la población "viven.en su entorno e interactúan entre ellos de forma descentralizada y distribuida. La fuente de inspiración original de los autores fueron los experimentos de vida artificial, en los que se define un escenario que recrea las condiciones biológicas de algún ecosistema, se sitúan en él componentes biológicos simples (células,

moléculas, etc) y también se definen reglas de interacción y supervivencia entre estos componentes. La evolución en los sistemas de Vida Artificial es open-ended, es decir, no tiene un criterio de parada predefinido, y el sistema se adapta de manera continua a los cambios en el entorno. Todas estas características llevaron a Watson et al. a proponer un algoritmo evolutivo similar pero que pudiese ser aplicado a la evolución de controladores para equipos de robots, de forma que la evolución pudiese llevarse a cabo en tiempo real y dentro de cada robot en un cierto entorno real.

Así, la definición de Embodied Evolution como algoritmo para la optimización de sistemas multi-robot sería la siguiente:

estrategia evolutiva basada en la evolución natural en la que los individuos que constituyen la población están embebidos en el cuerpo del robot y situados en un entorno en el que interactúan de manera local, asíncrona y descentralizada.

Estas interacciones no están guiadas por ningún mecanismo de sincronización preestablecido como en los algoritmos evolutivos tradicionales, sino que resultan de los comportamientos individuales de los robots y pueden cambiar en función de las condiciones de dicho entorno. Los individuos tienen un tiempo de vida máximo y cuando lo agotan, desaparecen y son reemplazados por sus descendientes. La evolución en Embodied Evolution es de final abierto (open-ended), lo que constituye un paradigma intrínsecamente adaptativo y altamente adecuado para resolver problemas multi-robot en tiempo real. Esta aproximación tiene un funcionamiento similar a la evolución que se produce en una ecología, es decir, existe un entorno dado, unos "seres"que lo habitan y unas reglas de interacción y relación. El sistema evoluciona hacia algún estado estable en el que las entradas y salidas de recursos se compensan, o bien el sistema colapsa y se extinguen los seres. En el caso que nos ocupa, el sistema debería tender hacia un estado estable que optimizase la función de calidad colectiva que se defina.

Detrás de esta definición básica existen numerosos aspectos que tratar con más detalle, como se hará en el próximo capítulo donde se definirá el algoritmo concreto que se ha desarrollado en esta tesis. Pero como se puede observar, es una aproximación que cumple con los requisitos establecidos con anterioridad. Por un lado, la evolución se produce en tiempo real (on-line). Por otro, permite obtener equipos genéticamente heterogéneos, de hecho, la aparición de especies es algo intrínseco a la evolución natural. Además, como se ejecuta dentro de cada robot (on-board), se cumple con la característica buscada de alto grado

Tabla 4.1: Trabajos más relevantes en Embodied Evolution

Embodied evolution	
Distribuida	Encapsulada
Watson (1999, 2002)	Takaya (2003)
Simoes (1999)	Walker (2003)
Nehmzow (2002)	Elfwing (2005, 2011)
Bianco (2004)	ALF Perez (2008)
Wischmann (2007)	Dong-Wook Lee (2008)
König (2008)	Eiben (2010)
Prieto (2010)	Haasdijk (2010)
Bredeche (2010, 2012)	O'Dowg (2011)
Karafotias (2011)	Schwarzer (2011)
Haasdijk (2014)	Heinerman (2016)

de autonomía de cada uno de los robots. Finalmente, Embodied Evolution es tolerante a cambios tanto en la composición del equipo, como en el entorno y en la propia tarea. En la siguiente sección se realizará una revisión de los principales trabajos que han seguido esta aproximación en los últimos años:

4.2. Embodied Evolution

En este apartado se revisarán los principales trabajos realizados dentro del paradigma de Embodied Evolution (EE), y que son los mostrados en la tabla 4.1. Comenzando por el mencionado trabajo de Watson et al. [Watson et al., 1999, Watson et al., 2002], donde presenta el concepto global del paradigma y una implementación concreta, el algoritmo PGTA (Probabilistic Gene Transfer Algorithm). Este algoritmo está basado en el algoritmo genético microbiano de Harvey, que se fundamenta en reescribir porciones del genotipo de un individuo de mala calidad con porciones de un genotipo de un individuo de mayor calidad. En este caso, se ha adaptado este proceso para realizarlo de forma descentralizada y distribuida. La transferencia se realiza de robot a robot, cuando ambos se encuentran en el escenario en un rango local, ya que cada uno de ellos emite información genética a una frecuencia proporcional a su nivel de energía virtual, que describe el rendimiento del robot en la tarea. Los genes que se emiten en la transferencia son seleccionados al azar, y si un robot está en rango de recibir

Figura 4.1: Fotografía del experimento de Watson et al. Fuente: "Embodied evolution: embodying an evolutionary algorithm in a population of robots" [Watson et al., 1999]

esta transmisión, puede aceptar genes con una probabilidad inversamente proporcional a su nivel de energía. De esta forma, los robots con mayor nivel de energía emitirán más genes y aceptarán menos genes de otros robots, mientras que los robots con menos energía emitirán muchos menos genes pero aceptarán con mayor probabilidad los de otros robots.

La tarea original propuesta en el trabajo de Watson et al. consiste en ir hacia una fuente de luz con ocho robots. La luz está situada en el centro del entorno y los robots deben moverse desde sus posiciones hasta ella. La figura 4.1 muestra una fotografía de los robots en el entorno. Como es habitual en robótica evolutiva, el controlador de los robots es una red de neuronas artificiales pero tienen dos comportamientos predefinidos que funcionan al margen de la red. Por un lado un comportamiento de reinicio, que se inicia una vez que el robot llega a la luz, para situar el robot en una posición y orientación aleatoria del entorno y volver a comenzar la tarea, y por otro lado, un comportamiento de giro aleatorio cuando el robot detecta que puede estar atrapado. Los robots ganan energía cada vez que se acercan a la luz y consumen energía cada vez que transmiten un gen. El algoritmo se compara con una solución diseñada a mano y una solución con parámetros aleatorios de los controladores y se demuestra que se pueden producir comportamientos de calidad con un algoritmo evolutivo descentralizado y asíncrono en una población de robots. El punto débil de este primer trabajo es que la tarea es muy simple y no es colectiva, sino que es una

tarea que podría realizar un robot pero se ha llevado a cabo con un sistema multi-robot.

Desde el trabajo original de Watson, han surgido numerosas variaciones dentro del paradigma de Embodied Evolution. Eiben et al. propusieron una clasificación de este paradigma en base a tres tipos de Embodied Evolution: distribuida, encapsulada e híbrida [Eiben et al., 2010]. Por orden, podemos hablar primero de Distributed Embodied Evolution (DEE), en la que los operadores genéticos se ejecutan de forma distribuida entre robots, y que es la variante fiel a la idea original de Embodied Evolution. Por otro lado, tenemos la Encapsulated Embodied Evolution (EEE), una aproximación en la que cada robot lleva una población dentro de sí mismo y aislada del resto, y los operadores genéticos se aplican dentro de cada población. El tercer tipo, la aproximación híbrida, combina los mecanismos de la aproximación encapsulada y de la distribuida. Es similar a una aproximación encapsulada pero en la que las poblaciones se comparten información a través de la migración entre robots. En esta tesis, esta aproximación se considera un caso particular de aproximación encapsulada, ya que la única diferencia es la existencia de la migración. A continuación, se describirán algunos trabajos de Embodied Evolution tanto encapsulada como distribuida.

4.2.1. Embodied Evolution Encapsulada

En la figura 4.2 se muestra una representación del funcionamiento de la aproximación encapsulada, en el que se observa cada uno de los robots con sus respectivas poblaciones dentro, y con un genotipo activo, que se ha transformado a fenotipo, y es el que está tomando el control del robot en cada momento.

Uno de los primeros trabajos de Embodied Evolution encapsulada es el trabajo de Takaya et al. [Takaya and Arita, 2003]. En este algoritmo, cada robot contiene una población de genotipos y transmite y recibe dichos genotipos cuando se encuentra con otros robots en el entorno. Cada robot, además de la población activa, gestiona una cola de genotipos recibidos, a la espera de ser evaluados. A la hora de generar nuevos genotipos en la población, hay que seleccionar dos genotipos de la población para cruzarse y el nuevo genotipo se almacena en la cola de genotipos, al igual que los genotipos recibidos por otros robots. Para transferirse de la cola de genotipos a la población del robot, los genotipos deben obtener una calidad mayor que el peor genotipo de la población,

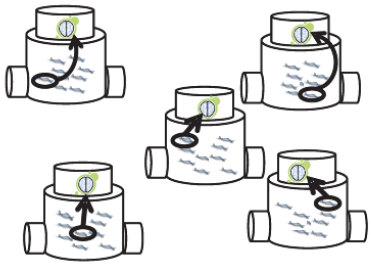


Figura 4.2: Esquema de Embodied Evolution Encapsulada en "Embodied, Online, On-board Evolution for Autonomous Robotics" [Eiben et al., 2010]

ya que de lo contrario se descartan. El controlador de cada robot es una red de neuronas artificiales codificada por estos genotipos. La tarea propuesta es una tarea de evitar obstáculos con seis robots Khepera simulados y se comparan diferentes configuraciones del algoritmo, con colas de genotipos de diferente tamaño y habilitando o no la migración de genotipos de una población a otra.

Otro ejemplo de Embodied Evolution encapsulada es el trabajo de Elfwing et al. [Elfwing et al., 2005]. Los autores argumentan que las aproximaciones distribuidas tienen una limitación que las hace difícil de aplicar que es el alto número de robots que necesitan. Para solucionar esto, proponen una población en cada robot y una estrategia de reparto de tiempo para evaluar la población en el robot. El reparto de tiempo consiste en dividir un tiempo de evaluación total por el numero de genotipos a evaluar y asignar este periodo a cada uno de ellos. En este caso concreto, el tiempo de evaluación de cada genotipo, corresponde a un cuarta parte de la vida del robot. Cuando se cumple este tiempo de vida, se reemplaza el controlador seleccionando uno de entre los genotipos recibidos en tiempo de vida según su calidad. La tarea planteada se basa en sobrevivir en un entorno desconocido, en la que los robots se deben buscar para cruzarse y deben encontrar baterías por el entorno para recargar la suya para no quedarse sin energía. En la figura 4.3 se muestra una fotografía del experimento. Los robots tienen aprendidos una serie de comportamientos básicos: buscar baterías, reproducirse, evitar obstáculos, y se evoluciona un controlador que es una red de neuronas que tiene que seleccionar entre estos comportamientos. El punto fuerte de esta aproximación es que se consiguen evolucionar comportamientos con tres robots reales, debido al encapsulamiento de las poblaciones en cada uno de ellos. Se compara esta aproximación encapsulada contra un algoritmo genéti-

co centralizado y los resultados obtenidos son prometedores ya que se obtienen mayor tasa de comportamientos de reproducción con Embodied Evolution encapsulada. Un punto débil de este trabajo, es que la tarea es simple ya que solo se tiene en cuenta capturar baterías y reproducirse.

Figura 4.3: Fotografía del experimento de Elfwing et al. en "Biologically Inspired Embodied Evolution of Survival" [Elfwing et al., 2005]

La mayoría de algoritmos utilizados en Embodied Evolution son algoritmos genéticos, pero también existen ejemplos en donde el método utilizado es programación genética. Un ejemplo de aproximación encapsulada con programación genética es el trabajo de Perez et al. [Perez et al., 2008]. En este trabajo, se evolucionan los controladores de cinco robots Khepera que deben realizar una tarea de navegación evitando obstáculos. Cada robot tiene una población de programas y las transmisiones entre ellos son porciones de estos programas. El proceso evolutivo en cada robot tiene en cuenta su población local y las partes recibidas de otros robots. Se demuestra que los robots aprenden a navegar el entorno evitando obstáculos. Un punto débil de este trabajo es que solo se realizan experimentos en simulación, pero hace una aportación, que es demostrar que es posible usar programación genética de una forma distribuida siguiendo el esquema de Embodied Evolution para optimizar un sistema multi-robot.

Dentro de esta aproximación, existen trabajos en los que se combina el algo-

ritmo evolutivo con un algoritmo de aprendizaje por refuerzo, como por ejemplo en [Walker et al., 2003]. En este trabajo se clasifican las metodologías para aplicar a la optimización multi-robot en dos tipos de métodos: métodos de evolución en fase de aprendizaje y métodos de adaptación en tiempo de vida por evolución. Se revisan trabajos en cada uno de estos métodos y se propone una nueva aproximación que combina estos dos métodos. La metodología propuesta se divide en dos fases diferentes: una fase de entrenamiento en la que se realiza evolución off-line y una fase de aprendizaje en tiempo de vida de los robots a través de evolución on-line. De este modo, la evolución off-line se aplicaría con el fin de obtener controladores robustos y genéricos y la evolución on-line para mejorar el rendimiento de los controladores. Para la evolución en tiempo de vida, se utiliza una estrategia evolutiva en la que se usa solo la mutación y cada robot lleva una población de tres cromosomas. La tarea que se resuelve es similar a la de Watson ya que consiste en seguir una fuente de luz evitando obstáculos. Se concluye que se mejora el rendimiento de los robots al aplicar evolución y aprendizaje en lugar de solo evolución. La aportación de este trabajo es que demuestra que aplicar aprendizaje en tiempo de vida ademas de evolución mejora el comportamiento del sistema multi-robot.

Otro ejemplo que combina Embodied Evolution con un algoritmo de aprendizaje por refuerzo es [Elfwing et al., 2011]. En anteriores trabajos de los autores [Elfwing et al., 2005], los comportamientos básicos de los robots estaban ya aprendidos antes de realizar la evolución, pero en este caso, el algoritmo de aprendizaje permite que los robots aprendan sus comportamientos dentro de cada generación del algoritmo evolutivo. El controlador de los robots es una red de neuronas que selecciona entre varias acciones básicas y el algoritmo evolutivo codifica además de los parámetros de la red, parámetros del algoritmo de aprendizaje. Los robots se intercambian genotipos a una frecuencia proporcional a su nivel de energía y estos genotipos se incluyen en las subpoblaciones de cada robot. La tarea en la que se analiza el algoritmo es una tarea de supervivencia con cuatro robots Cyber Rodent simulados, en un entorno con una serie de baterías virtuales con las que el robot puede recargar su nivel de energía. Posteriormente, se analizan los resultados obtenidos con la evolución a en un entorno real con dos robots.

Otro ejemplo en el que se combina evolución con aprendizaje es [Lee et al., 2008]. En este trabajo, el algoritmo evolutivo distribuido hace mejorar la habilidad para aprender de los robots al intercambiarse material genético con otros

robots, concretamente, el genotipo codifica la tabla de valores Q cada par de estado y acción, que los robots aprenden individualmente y se transmiten entre sí a través de comunicaciones entre ellos. Estas comunicaciones permiten realizar operadores de cruce entre soluciones nuevas con soluciones que ya han aprendido en el entorno, de forma que los individuos aprenden indirectamente a través de esto. Los genotipos solo se modifican cuando el robot que transmite tiene mayor calidad que el robot que recibe. La tarea que se resuelve es una tarea cooperativa de buscar y recolectar objetos por el entorno, evitando los obstáculos y las colisiones con otros robots. Existen cinco acciones predefinidas, que son: movimiento aleatorio, movimiento hacia delante, giro a la izquierda, giro a la derecha, y movimiento hacia objetivo. Se compara esta aproximación con respecto solo evolución o solo aprendizaje y este método obtiene el mejor rendimiento en la tarea. Se concluye que el método es muy efectivo ya que se obtiene mejor rendimiento en la tarea que en el caso de aplicar solo el algoritmo evolutivo distribuido. La aportación de este trabajo es demostrar la mejora el rendimiento en la tarea al combinar el aprendizaje de los robots con la evolución on-line, aunque solo en una tarea en simulación y no en robots reales.

Uno de los problemas de los algoritmos de Embodied Evolution encapsulada es que requieren una estrategia de reparto de tiempo para evaluar a todos los controladores de la población de cada robot, pero existen alternativas como por ejemplo el trabajo de O'Dowd [O'Dowd et al., 2011], en el que cada robot lleva un simulador embebido para evolucionar controladores antes de que tomen el control del robot y a mucha más velocidad que en el robot real, ya que se podría evaluar toda la población en el mismo tiempo que en el que se evalúa un solo controlador en el robot real. Con este método se introduce otra vez el problema de la transferencia al mundo real, ya que no se conoce el rendimiento de los controladores fuera del simulador. Este trabajo en particular se centra en buscar discrepancias entre los dos entornos, simulado y real. El algoritmo tiene dos componentes evolutivos: por una parte, un componente distribuido en todo el equipo de robots, en donde se evolucionan en tiempo real los modelos del entorno y se comunican entre robots, y por otra parte, un componente local de cada robot, con el que se evolucionan los controladores, en un tiempo mas rápido que el otro componente. Es decir, se evolucionan los simuladores de forma distribuida y los controladores de forma encapsulada en cada robot. La tarea resuelta es una tarea de encontrar y transportar comida a una ubicación realizada por 10 robots e-puck en la que los robots no pueden detectar la comida pero sí pueden detectar una fuente de luz. En la figura 4.4 se muestra una fo-

tografía de los robots en el experimento. El controlador de los robots selecciona comportamientos según el estado actual y existen cuatro comportamientos básicos preprogramados: evitar obstáculos, búsqueda aleatoria, moverse hacia luz, alejarse de la luz. Se muestra que el número de elementos de comida recolectada aumenta a lo largo de la evolución, con lo que se producen comportamientos eficientes en la tarea. La aportación de este trabajo es que solucionar el problema del largo tiempo requerido para evaluar una población que tienen las aproximaciones encapsuladas.

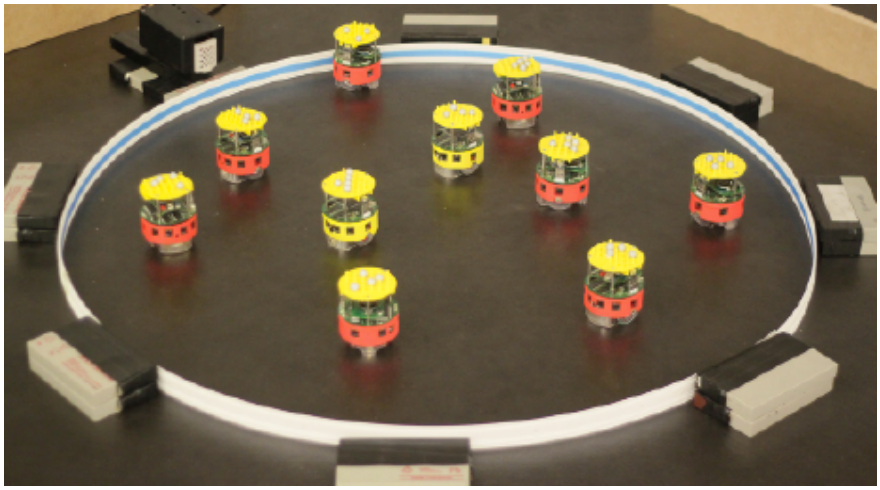


Figura 4.4: Fotografía del experimento de O'Dowd et al. en "The distributed coevolution of an embodied simulator and controller for swarm robot behaviours" [O'Dowd et al., 2011]

El algoritmo más relevante de Embodied Evolution encapsulada es el algoritmo (μ+1)-ONLINE [Eiben et al., 2010]. Es el más relevante ya que es el que más se ha usado en la bibliografía, por ejemplo en [Haasdijk et al., 2010], incluso con ejemplos recientes [Heinerman et al., 2016]. Este algoritmo está basado en las estrategias evolutivas (μ+1) en el que cada robot tiene encapsulada una población de μ individuos. Si μ es 1, el algoritmo queda como (1+1)-ONLINE y es una estrategia evolutiva en la que solo se aplica el operador de mutación. El tamaño de esta mutación está regulado por un parámetro que se adapta a lo largo de la evolución. La característica básica de este algoritmo es que los hijos solo reemplazan a su padre si su calidad es mayor. El tiempo de evaluación para

cada individuo de la población es fijo y cuando se acaba, otro controlador toma el control del robot en la posición en la que quedó el robot. Para poder recuperarse de un mal comportamiento del anterior controlador, se incluye un periodo de recuperación en el que no se computa la calidad al nuevo controlador. Con el fin de evitar evaluaciones que pudieron ser ruidosas por las condiciones que tuvieron los controladores, el mejor genotipo se reevalúa con una probabilidad. Se analiza el algoritmo con diferentes tamaños de población (μ=1,3,9,13) en una tarea de moverse lo más rápido y recto posible en varios entornos con diferentes obstáculos y se observa que aumentar el tamaño de las poblaciones incrementa el rendimiento del algoritmo en la tarea.

Un trabajo reciente que aplica el algoritmo (1+1)-ONLINE es el trabajo de Heinerman et al. que resuelve una tarea de recolección de comida con un equipo de robots Thymio II [Heinerman et al., 2016]. En la figura 4.5 se muestra una fotografía del experimento. Se comprueba que se aumenta el número de elementos de comida recolectados a lo largo del tiempo. La aportación de este trabajo es que es un experimento asequible, realizado con robots reales y en un tiempo muy limitado, ya que en una hora se obtienen los comportamientos óptimos, al contrario que en otros trabajos de evolución on-line en los que se requieren varias horas o días de evolución (por ejemplo, los experimentos originales de Watson et al. duraban 140 minutos).

Otro trabajo derivado del algoritmo (μ+1)-ONLINE es el trabajo de Schwarzer et al. [Schwarzer et al., 2011]. El esquema del algoritmo es igual al del algoritmo (μ+1)-ONLINE pero la novedad es que se permite la evolución estructural del genotipo y se utiliza evolución incremental complicando progresivamente la tarea para que los robots gradualmente adapten su comportamiento. El genoma codifica una red de neuronas en base a dos tipos de gen: gen de nodo y gen de enlace. El gen de nodo codifica una neurona de la red y el gen de enlace codifica una conexión en la red. De esta forma, se pueden codificar diferentes morfologías de la red al igual que se realiza en el algoritmo NEAT [Stanley and Miikkulainen, 2002]. La aportación de este trabajo es que se evoluciona la estructura del controlador de forma similar al NEAT pero de forma on-line y distribuida, lo cual lo hace más aplicable que el NEAT a sistemas multi-robot reales y ademas se hace utilizando evolución incremental, que es una metodología que se ha comprobado que funciona, para que la evolución progrese en tareas cada vez más complejas [Gomez and Miikkulainen, 1997].

Después de describir algunos de los trabajos más relevantes de Embodied

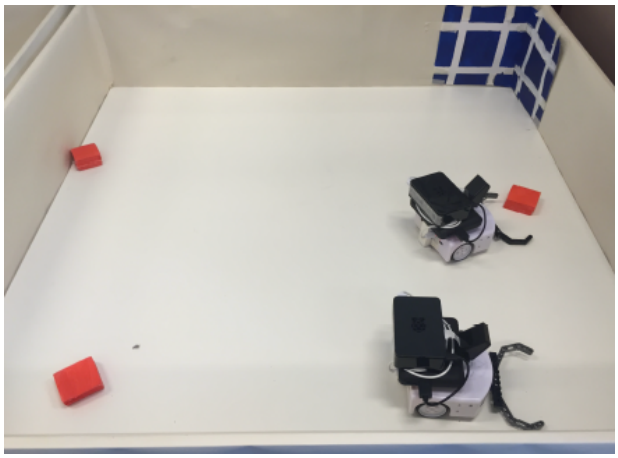


Figura 4.5: Fotografía del experimento de Heinerman et al. en "On-line Evolution of Foraging Behaviour in a Population of Real Robots" [O'Dowd et al., 2011]

Evolution Encapsulada, se puede concluir que una desventaja de estas aproximaciones es tener que repartir el tiempo de evaluación entre todos los controladores de una población, aunque esto tiene la ventaja de que debido a que encapsulan una población, la variabilidad genotípica es mayor que en un algoritmo distribuido. Esto permite utilizar la aproximación encapsulada en equipos de muy pocos robots. Por ejemplo, en los trabajos de Elfwing se utilizan grupos de 4 o menos robots reales y esto no sería posible con un algoritmo distribuido porque se requiere mayor variabilidad genotípica.

4.2.2. Embodied Evolution Distribuida

En la figura 4.6 se muestra una representación del funcionamiento de la aproximación distribuida. Cada robot en lugar de una población dentro de sí, tiene únicamente un genotipo que es el que está controlando a su vez el robot. Los robots se transmiten entre sí estos genotipos mediante alguna estrategia de comunicación local.

Los algoritmos de Embodied Evolution distribuida (DEE) son fieles al esquema del trabajo original de Watson. Uno de los primeros ejemplos en esta

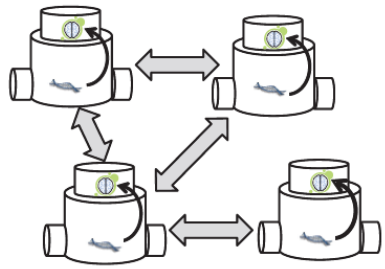


Figura 4.6: Esquema de Embodied Evolution Distribuida en "Embodied, Online, On-board Evolution for Autonomous Robotics" [Eiben et al., 2010]

aproximación es el Evolutionary Control System de Simoes et al. [Simoes and Dimond, 1999], que también realiza una evolución distribuida sobre un equipo de cinco robots. El funcionamiento de este algoritmo es diferente al de Watson porque está formado por dos períodos: un periodo de operación para evaluar controladores y un periodo de recombinación que al terminar inicia el periodo de operación de nuevo. En el periodo de recombinación, los robots envían peticiones de recombinación a otros y se generan nuevos controladores. El robot con mayor calidad pasa a la siguiente generación sin necesidad de recombinarse. La tarea resuelta en este caso es una tarea de navegación en un entorno con obstáculos, y el controlador de cada robot es una red de neuronas artificiales que debe decidir entre 8 comandos a cada motor. En el genotipo, además de codificarse los parámetros de la red, se evolucionan también características de los robots, como el uso o no de un sensor y parámetros de velocidad de los comandos. El algoritmo se prueba en diferentes configuraciones del entorno, introduciendo más obstáculos y paredes verticales en él. Los resultados obtenidos demuestran que la técnica es capaz de evolucionar los controladores de los robots para que realicen la tarea deseada y se muestra una técnica prometedora en la adaptación a un entorno complejo y ruidoso.

Otro trabajo similar al algoritmo PGTA, pero en el que se transmiten además de porciones del código genético, calidades, es el de algoritmo PEGA (Physically Embedded Genetic Algoritm) [Nehmzow, 2002]. En este algoritmo, los robots llevan dos genotipos consigo: el genotipo activo y el genotipo que ha obtenido mayor nivel de calidad hasta el momento. Este algoritmo puede no considerarse un algoritmo de Embodied Evolution, ya que el proceso de reproducción está

prefijado para que suceda después de que los individuos sean evaluados, pero por su similitud a un algoritmo de Embodied Evolution, se describirá aquí. Los robots evalúan un genotipo durante un tiempo establecido y posteriormente buscan en el entorno a otro robot para poder intercambiarse información genética. Cuando se encuentran en un rango de comunicaciones, se intercambian su código genético actual y su nivel de calidad. Para un cruce, si el código genético recibido tiene mayor nivel de calidad que el propio, se reemplaza mitad del código genético propio con el recibido, mientras que si tiene menor nivel de calidad no se reemplaza nada. Las mutaciones ocurren inversamente proporcionales al nivel de calidad. La tarea que se propone es una tarea que necesita realizar tres comportamientos: acercarse a la luz, evitar obstáculos y buscar otros robots. El controlador de los robots selecciona entre diferentes acciones predefinidas según las entradas de sus sensores. Este algoritmo es otra comprobación de que se puede realizar optimización en tiempo real de un sistema multi-robot a través de, en este caso, un algoritmo genético distribuido dentro de los robots.

Otro trabajo muy similar es el de Wischmann et al. [Wischmann et al., 2007]. En este trabajo se aplica el algoritmo PGTA de Watson para estudiar el efecto del tiempo de maduración, que es un periodo que asegura un desarrollo mínimo de los individuos sin tener presión selectiva, en un ejemplo de depredador presa. Existe una población de ovejas y una población de lobos, que en este caso no evolucionan para no tener el efecto de la Reina Roja descrito previamente en el anterior capítulo. La novedad de este algoritmo, con respecto al de Watson, es la introducción de este periodo de maduración, en el que el robot no transmite genes ni recibe los de otros robots para no sobreescribir los suyos prematuramente. Al igual que en el algoritmo de Watson, la evaluación de la calidad se representa a través de un nivel de energía que tiene cada robot. Las ovejas ganan energía cuando no hay un lobo en su área de colisión, y los lobos a su vez ganan cuando sí tienen ovejas en su área de colisión. Realizar una transmisión de genes tiene un coste de energía. Se demuestra que el establecimiento de un tiempo de maduración correcto afecta si el aprendizaje favorece a la evolución. Esta conclusión sobre el tiempo de maduración se tendrá en cuenta en esta tesis ya que también se definirá un tiempo de maduración en el algoritmo desarrollado en ella.

En Embodied Evolution encapsulada se han descrito algunos trabajos en los que se aplica programación genética en lugar de un algoritmo genético. Un ejemplo de esto en Embodied Evolution distribuida es el trabajo de König et

al. [König et al., 2008]. En este trabajo se plantea una tarea de evitar colisiones entre un equipo de veinte micro robots Jasmine IIIp. En la figura 4.7 se muestra una fotografía de los robots en el experimento. Los controladores de los robots son autómatas y los genotipos codifican unas estructuras llamadas fenomas que a su vez generan los autómatas. En este caso, los operadores genéticos se aplican sobre los fenomas, que también es lo que se transmiten los robots cuando se encuentra en rangos locales. Se analiza el algoritmo para comprobar si se pueden conseguir resultados estables en la tarea pero se observa que algunos robots no tienen comportamientos de evitar obstáculos eficientes.

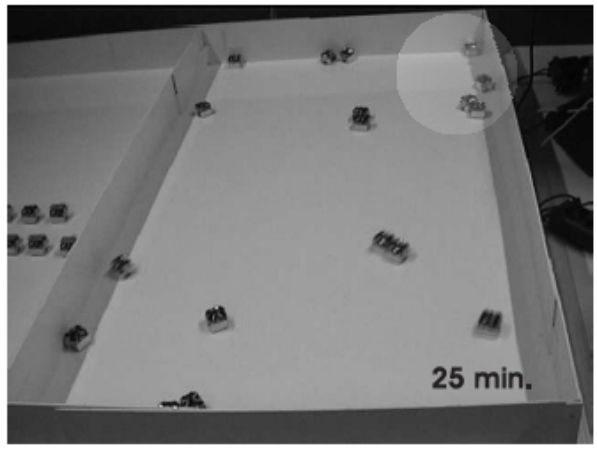


Figura 4.7: Fotografía del experimento de König et al. en "Stability of On-Line and On-Board Evolving of Adaptive Collective Behavior" [König et al., 2008]

Otro ejemplo de Embodied Evolution distribuida es el de Bianco y Nolfi [Bianco and Nolfi, 2004]. Los autores plantean un experimento con un equipo de robots en simulación en el que no se resuelve una tarea sino que se realiza una evolución "open-ended" para analizar los comportamientos que emergen como buscar y perseguir a otros robots. De forma similar a la evolución natural, en este experimento no se introduce ningún criterio de calidad explícito sino que lo que habilita la reproducción es la propia capacidad de hacerlo. Los robots son capaces de conectarse físicamente a otros robots, y al hacer esto, uno de ellos es el que toma el control de la unión, reemplazando el genotipo del otro robot para formar una nueva unidad robótica. A su vez estas nuevas unidades se pueden

unir a otras formando agregaciones de diferente tamaño. Con este experimento se comprueba que unas reglas simples pueden llevar a cabo un proceso de evolución en el que se crean necesidades adaptativas (como coordinarse y cooperar) y se mejoran las soluciones a estas necesidades.

El algoritmo ASICo es un algoritmo de Embodied Evolution distribuido desarrollado por el GII de la UDC [Prieto et al., 2010] y en el que la evolución está guiada por la energía. Los robots ganan o pierden energía según sus interacciones en el entorno. Cada robot lleva consigo un genotipo activo que es el que controla al robot y un genotipo embrión, que es el que tomará el control del robot cuando se reemplace al genotipo activo. Los reemplazos se producen cuando el robot pierde toda su energía o sobrevive un tiempo máximo. Las transmisiones de códigos genéticos se producen cuando dos robots se encuentran en un rango local de comunicaciones y se comunican su genotipo y su nivel de energía. El genotipo recibido en esta transferencia será o no aceptado para modificar el embrión del robot según un criterio de selección, que usualmente es que tenga mayor nivel de energía. El algoritmo ASiCo ha sido aplicado a problemas de optimización distribuidos, como optimización de fletes de buques [Prieto et al., 2011] y también a tareas colectivas en sistemas multi-robot, como tareas de limpieza colectiva [Prieto et al., 2010], de búsqueda y recolección colectiva o de vigilancia. En la figura 4.8 se muestra una fotografía de los robots en el experimento de limpieza colectiva.

Un ejemplo de tarea [Trueba et al., 2013] se centra en la limpieza colectiva por 9 robots e-puck, en la que los robots deben buscar bloques en el entorno y unirlos con otros bloques para formar parejas de bloques que se transportan hacia fuera del entorno. En primer lugar, tienen que marcar bloques como listos para recogida. A continuación, se deben formar parejas de bloques listos para recogida. Cada vez que se unen dos bloques listos para recogida, se transportan automáticamente hacia fuera del entorno. Los bloques no preparados para recogida tienen color gris y los bloques listos para recogida tienen color azul, para que los robots puedan diferenciarlos con sus cámaras. El objetivo de la tarea es maximizar el número de bloques recogidos. Debido a que la tarea tiene dos subtareas diferenciadas, buscar bloques no preparados y marcarlos como listos, y por otro lado buscar bloques listos para recogida y transportarlos hacia otros bloques listos, la tarea puede ser resuelta de forma más eficiente a través de especialización, pero hay que tener en cuenta que en esta tarea la especialización no tiene una separación espacial, es decir, los individuos especializados

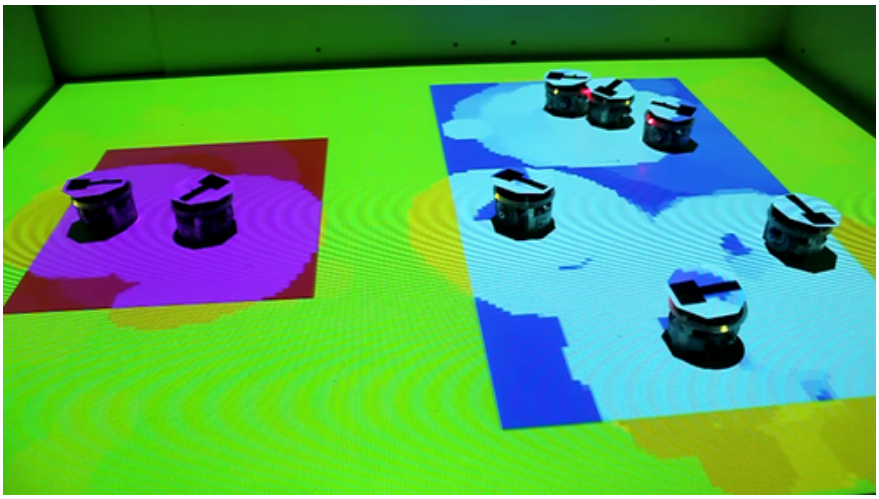


Figura 4.8: Fotografía del experimento de Prieto et al. en "Open-ended evolution as a means to self-organize heterogeneous multi-robot systems in real time" [Prieto et al., 2010]

no se sitúan en áreas espaciales del entorno según su especie, sino que en esta tarea todas las especies robots interactúan por todo el entorno. Los resultados demuestran que, a pesar de no estar separadas en el entorno las especies, estableciendo un buen balance entre exploración y explotación, el algoritmo ASiCo permite obtener especialización.

Un algoritmo relevante de Embodied Evolution distribuida es el algoritmo mEDEA (Minimal Environment Driven Evolutionary Algorithm) [Bredeche and Montanier, 2010, Bredeche et al., 2012]. El algoritmo mEDEA busca que los robots se adapten al entorno y sobrevivan, sin una función de calidad explícita, sino de forma implícita a través del entorno. Los robots transmiten su código genético en un rango local y tienen una lista de genotipos importados recibidos de otros robots. La selección no opera a nivel local de individuo, ya que se selecciona un genotipo aleatorio de la lista de genotipos importados, sino a nivel global, ya que los genotipos más transmitidos serán más seleccionados. Los robots tienen un nivel de energía y solo se mueven y transmiten su código genotipo cuando su energía es positiva mayor que cero. Cuando su energía llega a cero, se quedan inmóviles en el entorno hasta completar su tiempo de vida, que es fijo. Al completar este tiempo, se produce un reemplazo seleccionando

un genotipo aleatoriamente de su lista de genotipos importados.

Se estudia el algoritmo en un experimento con 100 robots e-puck simulados y con dos configuraciones del entorno: una configuración de movimiento libre con obstáculos en el entorno en el que los robots no pierden energía al moverse y una configuración en la que los robots pierden una unidad de energía al moverse, pero pueden ganarla al recolectar trozos de comida situados en el entorno. El experimento consiste en cambiar de una configuración a la otra de forma abrupta y se observa que a pesar del cambio en el que muchos robots quedan inactivos, el algoritmo consigue propagar individuos con calidad y se vuelve a aumentar el número de robots activo, con lo que se puede concluir que el algoritmo es robusto a cambios en el entorno. Otro experimento planteado es un entorno en el que se incluye un sol en un lado. Los robots no consumen energía en este experimento y el sol no produce ninguna ventaja de energía. Cuando se llega a un número determinado de generaciones, el sol aparece en el lado contrario del entorno. Se demuestra que los robots llegan al consenso de ir hacia el sol, ya que esta estrategia les maximiza las oportunidades de propagar sus códigos genéticos. El algoritmo también se ha validado en robots reales, en concreto, se realizaron experimentos con equipos de entre 9 y 20 robots e-puck, en un entorno en el que se sitúa un objeto haciendo de sol como en el experimento anterior [Bredeche and Montanier, 2010]. En la figura 4.9 se muestran varias fotografías de los robots en el experimento. Se comprueba que existen muchas diferencias entre la simulación y los robots reales debidos pero que el algoritmo mEDEA es robusto a esto, ya que en robots reales también es capaz de evolucionar comportamientos similares a los obtenidos en simulación.

Un algoritmo derivado del algoritmo mEDEA es el algoritmo MONEE [Haasdijk et al., 2014], que introduce la presión de la tarea a la presión ejercida por el entorno del mEDEA. Esta presión de la tarea se obtiene de obtener créditos de realizar acciones en el entorno. Cuando existen múltiples subtareas, se cuentan por separado los créditos para cada una de estas subtareas. La reproducción en este algoritmo es diferente a la forma de reproducción del mEDEA, ya que en este caso los robots no reciben transmisiones de códigos genéticos hasta que están en un estado de incubación, en el que están inmóviles en el entorno recibiendo estas transmisiones. En las transmisiones se incluyen los créditos de los genotipos, de forma que cuando acaba esta fase de incubación, se obtiene un código genético con una selección basada en ranking teniendo en cuenta los créditos de cada tarea y la escasez de los créditos de cada tarea, con el fin pa-

Figura 4.9: Fotografía del experimento de Bredeche et al. en "Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents" [Bredeche et al., 2012]

ra puntuar menos las tareas más fáciles de realizar (más créditos disponibles) con las difíciles. De esta forma la selección se convierte en un mercado con los comportamientos que se necesitan. El algoritmo se analiza en una tarea de recolección colectiva con 100 robots e-puck simulados en un entorno con obstáculos. Existen dos tipos de objetos, cada tipo de un color, para recolectar en el entorno. Recolectar un tipo de objeto supone una tarea diferente a la de recolectar el otro tipo y se debe evolucionar por separado. Se compara el algoritmo con la selección de mercado o sin ella y con el algoritmo mEDEA. Debido a que en el algoritmo mEDEA no se da crédito por recolectar objetos, en este caso, realiza un comportamiento aleatorio en la tarea. Se comprueba que el algoritmo MONEE con selección de mercado es el que mejor resuelve la tarea.

Otro trabajo relevante donde se compara un algoritmo de Embodied Evolution distribuida con uno Embodied Evolution encapsulada es el trabajo de Karafotias et al. [Karafotias et al., 2011]. Aquí se presenta un nuevo algoritmo distribuido de embodied evolution, denominado EDEA (Embodied Distributed Evolutionary Algorithm) (no confundir con mEDEA) y se compara con el algoritmo (μ+1)-ONLINE encapsulado. El funcionamiento del algoritmo es similar al algoritmo de Watson et al., pero en este caso los robots tienen un rango de comunicaciones global y pueden contactar con cualquier robot para transmitir información. Se analiza el algoritmo en varias tareas como una de moverse hacia una luz, una de moverse lo mas rápido y recto posible, y una tarea de vigilancia colectiva. A diferencia de las otras dos, en la tarea de vigilancia colectiva se

requiere coordinación. En esta última, los robots dispersan feromonas por el entorno que otros robots pueden detectar. El algoritmo EDEA obtiene mejor rendimiento en dos de las tres tareas más simples: la de seguir la luz y moverse lo mas rápido posible, mientras que el algoritmo (μ+1)-ONLINE obtiene mejor rendimiento en la tarea de vigilancia colectiva. Una posible causa de eso es que la tarea penaliza comportamientos similares ya que los robots siguen los mismos caminos, y el algoritmo (μ+1) obtiene comportamientos más diversos.

Una vez descritos los trabajos más relevantes de la aproximación distribuida, se puede argumentar que esta aproximación tiene las ventajas de ser más escalable que la aproximación encapsulada a equipos de robots muy numerosos, y no tiene el problema de tener que repartir el tiempo de evaluación dentro de cada robot que sí tiene la aproximación encapsulada. Esto produce que también sea más simple en su funcionamiento que la encapsulada, permitiendo robots más simples, lo cual la hace más apropiada para la aplicación en un sistema multi-robot real. Muchos de los trabajos en la aproximación distribuida son muy similares conceptualmente pero con particularidades de diseño propias, que hace difícil escoger uno sobre otro. De ahí ha surgido el objetivo básico de esta tesis, la necesidad de generalizar este paradigma en un algoritmo canónico que sea más simple y estándar que los existentes. El siguiente capítulo contiene la definición de este algoritmo canónico y sus parámetros intrínsecos.

Capítulo 5

Algoritmo canónico

En este capítulo se describirá el algoritmo canónico de Embodied Evolution distribuido que se ha desarrollado en esta tesis. El capítulo se ha dividido en las siguientes secciones: análisis de otros algoritmos similares, extracción de procesos básicos, parámetros intrínsecos del algoritmo canónico y pseudocódigo del mismo.

5.1. Extracción de procesos básicos

El funcionamiento de las aproximaciones de Embodied Evolution (EE) es idéntico al funcionamiento de un algoritmo genético estándar, ya sea aplicado a la coordinación de robots o no, y se puede dividir en tres procesos esenciales: evaluación, reproducción y reemplazo. La reproducción, a su vez, consta de dos etapas que son la selección y recombinación con operadores genéticos. Como se explicó en el capítulo 3, en un algoritmo genético estándar, estos tres procesos esenciales conforman una generación del algoritmo, que se repite cíclicamente hasta completar las generaciones o pasos de evolución deseados. Se parte de una población de individuos que son las soluciones al problema, se evalúan con una función de calidad y se generan nuevas soluciones a través de la recombinación de algunos individuos seleccionados. Estas soluciones reemplazan otras, frecuentemente de menor calidad, y se repite este ciclo. En el caso de un algoritmo de EE distribuido (DEE) las procesos son idénticos, excepto que en este

caso ocurren de forma distribuida dentro del cuerpo de cada robot. Debido que se ejecutan de forma paralela en cada robot, se pierde la secuencia fija que existe en los algoritmo genéticos estándar y se intercalan estos procesos, surgiendo de las interacciones entre robots. A pesar de este carácter paralelo de los procesos en EE, los procesos básicos son los mismos con lo cual, si queremos generalizar el esqueleto del algoritmo canónico de DEE debe seguir la estructura de un algoritmo genético.

5.2. Análisis de otros algoritmos

Con el fin de generalizar el paradigma de DEE, se han analizado en detalle los algoritmos más relevantes en la literatura en cuanto al número de publicaciones científicas que ha generado: el algoritmo PGTA [Watson et al., 1999, Watson et al., 2002], el mEDEA [Bredeche and Montanier, 2010, Bredeche et al., 2012] y el ASiCo [Prieto et al., 2011, Prieto et al., 2010]. En la tabla 5.1 se muestra una comparativa de las características de estos tres algoritmos organizadas en los tres grandes procesos que ejecutan (evaluación, reproducción y reemplazo).

En cuanto al proceso de evaluación, los tres algoritmos están basados en una calidad individual representada en el robot a través de un nivel de energía, como es típico en los sistemas de vida artificial. Debido a que en estos algoritmos la evolución se realiza dentro del cuerpo del robot, la única forma de evaluar un controlador de robot sin recurrir a ningún elemento externo y manteniendo la característica de algoritmo distribuido sin elementos centrales, es que cada uno estime su propia calidad, es decir, conozca su propia energía. Los robots consumen energía en su interacciones pero también la pueden obtener en el entorno cuando realizan determinadas acciones recompensadas por la función de transferencia de energía. La función de transferencia de energía es una función de calidad diseñada para cada problema particular teniendo en cuenta que la maximización de la energía individual suponga la maximización de la función de calidad a nivel global. La energía individual puede ser calculada como energía instantánea, energía máxima o energía promedio en una ventana de tiempo sin que afecte al proceso de maximización a nivel global.

Con respecto al cálculo del valor de calidad, en robótica evolutiva se pueden dividir las aproximaciones en aquellas que utilizan una función de calidad explícita o implícita. Las funciones de calidad explícitas modelan no solo que el

Tabla 5.1: Análisis de otros algoritmos

	Evaluación	Reproducción			Reemplazo
		Tamaño máximo ventana de selección	Política de selección	Operadores genéticos	
PGTA	Calidad individual (energía)	1	Calidad individual	Como en Microbial GA	Inmediato en la reproducción
EDEA	Calidad individual (energía)	Tamaño de la población	Calidad individual	Mutación gaussiana	Síncrono en cada generación
ASiCo	Calidad individual (energía)	Ilimitada	Calidad individual	Mutación y cruce	Independiente de la reproducción

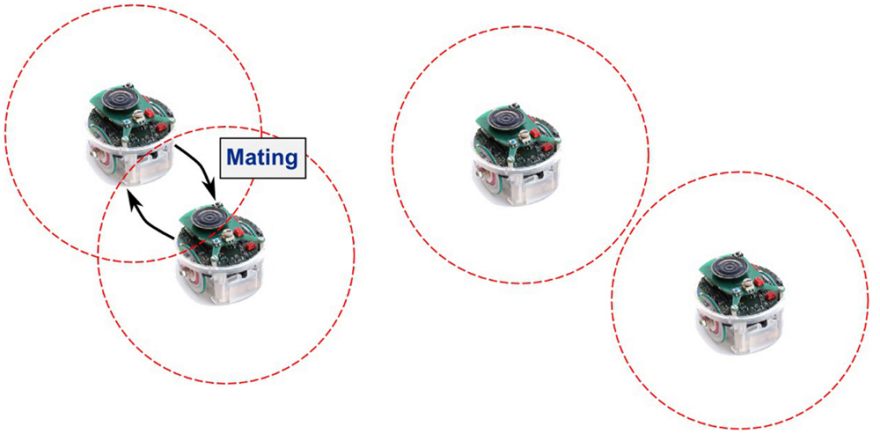


Figura 5.1: Esquema de reproducción en un algoritmo de EE. Figura de "Evolutionary robotics: what, why, and where to" [Doncieux et al., 2015]

robot obtenga un determinado comportamiento sino también la forma de obtenerlo, mientras que en las funciones implícitas se recompensa que se obtenga el comportamiento final, y se proporciona menos rigidez y más libertad al algoritmo de obtener el comportamiento deseado. Dentro de los trabajos de DEE, la tendencia es aplicar una función de calidad implícita. Un ejemplo de esta tendencia por funciones implícitas es el trabajo original del PGTA en una tarea de búsqueda de una fuente de luz [Watson et al., 1999]. En este trabajo, los robots solo obtienen energía cuando han alcanzado la luz, pero no la obtienen por estar en camino hacia la fuente de luz. En el algoritmo mEDEA también se han realizado ejemplos concretos con funciones implícitas, como por ejemplo en una tarea de búsqueda de comida en la que los robots solo obtienen energía cuando han alcanzado los trozos de comida [Bredeche and Montanier, 2010]. En el algoritmo ASiCo este tipo de función implícita se ha utilizado, por ejemplo, en una tarea de vigilancia colectiva [Prieto et al., 2010].

En cuanto a la reproducción, en este paradigma de EE distribuido las reproducciones son locales y asíncronas y están sujetas a que los robots se encuentren físicamente en el entorno (ver figura 5.2) o estén en un rango de comunicaciones local. La reproducción en estos algoritmos se puede dividir en tres procesos: en primer lugar, el robot que quiera reproducirse tiene que seleccionar un conjunto de robots como candidatos para reproducción, en segundo lugar, hay que esco-

ger uno de los candidatos, y finalmente, se aplican operadores de recombinación de genes con este candidato. Estos tres procesos se pueden caracterizar en tres parámetros:

1. Tamaño máximo de la ventana de selección
2. Política de selección
3. Operadores de recombinación genética

Con respecto el proceso de selección, en el caso de un algoritmo evolutivo tradicional se realiza en base a un torneo que incluye a un número fijo de individuos, el tamaño del torneo, y posteriormente se elige con mayor probabilidad los individuos de calidad más alta. En el caso del algoritmo PGTA, sin embargo, el tamaño de este torneo es 1, porque cada vez que un robot se encuentra a otro siempre se transmiten código genético. En el caso del algoritmo mEDEA, cada robot almacena una lista de genotipos recibidos y podría recibir tantos genotipos como toda la población en su tiempo de vida, con lo cual el tamaño máximo de esta ventana de selección es igual al número de robots. En el algoritmo ASiCo, el tamaño de esta ventana es ilimitado, ya cada robot almacena un embrión que se va modificando a medida que se encuentra con robots sin un límite establecido de encuentros (no existen generaciones y podría encontrarse con más genotipos diferentes que número de robots).

El segundo proceso de la reproducción es seleccionar uno de los individuos de entre los candidatos. La política de selección en los tres algoritmos va ligada a la calidad individual y la localización espacial de los robots. En el algoritmo PGTA, la calidad individual determina la frecuencia con la que un robot transmite sus genes. En el algoritmo mEDEA se transmite en un rango fijo y la selección es aleatoria dentro de la lista de genotipos importados, pero la calidad individual influye en el tiempo de vida de los robots y en su capacidad para propagar los genotipos. Los genotipos de mayor calidad individual se seleccionarán en mayor proporción porque son los más propagados en las listas de genotipos importados de los robots. De forma similar, en el algoritmo ASiCo el rango de transmisiones es fijo también pero se va seleccionando el individuo con mayor calidad individual, ya que solo se modifica el embrión si la estimación de calidad del nuevo embrión es mejor que la actual del embrión. En cuanto a los operadores genéticos, en el algoritmo PGTA se transmiten genes individuales a los que se les aplica una mutación, como en el algoritmo Microbial GA. En el algoritmo

EDEA, el operador está inspirado en las mutaciones de las estrategias evolutivas y por ello realiza una mutación gaussiana a cada uno de los genotipos en la lista de genotipos importados de cada robot cuando acaba cada generación. En el algoritmo ASiCo, se realizan operadores de mutación y de cruce bipolar en una proporción. El operador de mutación realiza una ligera variación sobre el genotipo sobre el que se aplica, mientras que el operador de cruce bipolar se aplica sobre dos genotipos para generar un descendiente con características de ambos padres.

Por último, en cuanto al tercer proceso de la reproducción, el reemplazo, los tres algoritmos de EE distribuido están basados en poblaciones fijas de robots, con lo que cada que reemplazo supone eliminar un código genético de uno de los robots. En el algoritmo PGTA el reemplazo nunca es un reemplazo de un código genético entero sino que son reemplazos parciales que se producen de forma instantánea en cada reproducción cada vez que se recibe un gen y se acepta para sobreescribir el gen actual. En el algoritmo mEDEA, las generaciones son síncronas, y una vez que el robot ha cumplido un tiempo de vida, se produce el reemplazo seleccionando un genotipo de la lista de genotipos importados. En el algoritmo ASiCo, los reemplazos se realizan independientemente de la reproducción, cuando un robot pierde toda su energía reemplaza su código genético actual por el de su embrión.

5.3. Parámetros intrínsecos definidos en el algoritmo canónico

Una vez analizados los procesos básicos y las características particulares de los algoritmos de DEE, se han modelado estos procesos mediante funciones de probabilidad genéricas, tratando de separar la influencia del entorno y de la tarea en el funcionamiento del algoritmo. Estos modelos probabilísticos se han parametrizado de la forma más simple y general posible de tal forma que la variación de estos parámetros tenga una influencia relevante en la operación del algoritmo. A continuación se describe esta parametrización estructurada según el proceso del algoritmo al que corresponde.

5.3.1. Evaluación

Este proceso no implica ninguna función genérica que lo modele ni tiene asociado ningún parámetro intrínseco en el algoritmo canónico, ya que no es generalizable y depende del problema a resolver. El único requerimiento para la evaluación en el algoritmo canónico es que se respeta la naturaleza distribuida de este tipo de algoritmo.

5.3.2. Reproducción

Como se ha descrito, la reproducción en los algoritmos de EE distribuido es local y asíncrona, y surge de los encuentros de los robots en el escenario. Existen tres aspectos de la reproducción que se generalizan con el algoritmo canónico de EE distribuido: el tamaño máximo de la ventana de selección, la política de selección y la probabilidad de aplicar unos operadores genéticos sobre otros.

5.3.2.1. Tamaño máximo de la ventana de selección

La reproducción está regulada por una distribución de probabilidad uniforme que determina la frecuencia con la que un robot entra en un modo de reproducción en cada paso de tiempo, que persigue que el número promedio de encuentros reproductivos para una ventana de tiempo T_{max} (tiempo de vida máximo), se corresponda con el parámetro S_{max} que representa el tamaño máximo de la ventana de selección. Por lo tanto, esta frecuencia se calcula con la siguiente expresión:

$$P_{mating} = \frac{S_{max}}{T_{max}} \tag{5.1}$$

Siendo S_{max} el tamaño máximo de la ventana de selección y (T_{max}) el tiempo de vida máximo de un robot. Una vez que el robot entra en este modo de reproducción, comprueba si entre los robots que están a su alcance hay alguno que esté también en este modo, permaneciendo en este modo si no es así hasta que lo haya. El tamaño máximo de la ventana de selección representa un límite teórico del máximo de robots que un robot encontraría disponibles para reproducción cuando viva hasta T_{max} (longevidad máxima). El tiempo de vida máximo (T_{max}) es fijo y solo afecta a la duración de las generaciones en la evolución. Por lo tanto, el primer parámetro intrínseco que afecta a la reproducción

es el tamaño máximo de la ventana de selección S_{max}.

5.3.2.2. Política de selección

En el algoritmo canónico se han establecido tres criterios de selección que generalizan los criterios que aplican los algoritmos de EE distribuido analizados antes: criterio basado en genotipo, criterio basado en un estado (por ejemplo basado en una posición espacial) o un criterio basado en calidad individual. Para el criterio basado en genotipo se puede definir una medida de similitud de genotipos (por ejemplo, la norma entre dos genotipos) y seleccionar individuos lo más similares posible en genotipo. Este criterio tiene utilidad desde el punto de vista de la especialización, ya que para mantener individuos especializados, se puede favorecer que un individuo siempre se reproduzca con otro individuo de su misma especie. Esto es lo que se aplica, por ejemplo, en el algoritmo CONE. En el criterio basado en estado, por ejemplo en posición espacial, se selecciona el robot más cercano al que realiza la selección, que para problemas en los que las sub-tareas están divididas en varias zonas del entorno puede ser beneficioso para la especialización, ya que los robots cercanos a un robot serán de su misma especie. El último de los criterios, el basado en calidad, es el más común en algoritmos genéticos tradicionales, usualmente aplicado con un torneo en el que seleccionarán con mayor probabilidad los individuos de mayor calidad. Este criterio tiene interés para converger lo antes posible hacia las soluciones de mejor calidad.

Todos estos criterios definen otro parámetro intrínseco del algoritmo que es la probabilidad que tiene un individuo de ser seleccionado, siguiendo la siguiente ecuación:

$$P_{eligibility} = \psi_e(\varphi_e, \gamma_e, [P_{cand}]) \tag{5.2}$$

La función ψ_e es la política de selección y decide en base a tres términos que son φ_e similitud de genotipos, γ_e calidad de genotipo, y P_{cand} posición del candidato, los criterios explicados anteriormente. Para el caso en el que sólo se considere un criterio basado en calidad, la función ψ_e solo tendría en cuenta el termino γ_e y daría 1 para el individuo con mayor calidad y 0 para el resto de individuos.

5.3.2.3. Operadores genéticos

En el algoritmo canónico de DEE, cuando se ha seleccionado un individuo para su reproducción, ya se puede aplicar el operador genético a sus códigos genéticos para generar un nuevo individuo. Los dos principales operadores genéticos en el campo de la computación evolutiva son el de mutación y el de cruce. El operador de cruce es similar a la reproducción sexual biológica, ya que necesita dos individuos que serán los padres y se utiliza información de ambos para generar un nuevo individuo que combina características de ambos. Sin embargo, para algunas codificaciones genotipo-fenotipo, el comportamiento final del nuevo individuo está frecuentemente muy alejado del de los padres, produciendo que haya muy poca correlación entre la calidad de los padres y la del nuevo individuo. Por otra parte, el operador de mutación produce una pequeña variación en el código genético, que implica una búsqueda local que explota las cercanías de las soluciones anteriores.

Para el algoritmo canónico, se ha definido un parámetro intrínseco que indica el balance entre cruce y mutación, y que en particular, indica la probabilidad de aplicar el operador de mutación P_{ls} y al mismo tiempo la probabilidad de cruce, que sería la complementaria ($P_{cr} = 1 - P_{ls}$). Esta probabilidad P_{ls} es una medida del balance entre exploración y explotación en el espacio de búsqueda a través de esta relación entre la frecuencia de la mutación (explotación) y la frecuencia del cruce (exploración).

La figura 5.2 muestra la distribución de probabilidad en la posición del nuevo gen si utilizamos una u otra técnica de reproducción. En el eje x, el gen "a" tiene un valor de 0.33 y el gen "b" 0.66. En el eje y se muestra la función de densidad de probabilidad de los operadores, fijados los parámetros de la mutación y el cruce, que regularían la anchura de estas áreas. Se puede observar que la probabilidad acumulada de aplicar una mutación, (suma de áreas) es P_{ls} mientras que la probabilidad del cruce es $1 - P_{ls}$. Es necesario puntualizar que esta representación se basa en operadores aplicados para genotipos formados por un solo gen, pero para genotipos formados por vectores de genes, en el caso de una mutación, se aplicaría una mutación a una porción del genotipo y el cruce realizaría un genotipo con porciones de los dos genotipos.

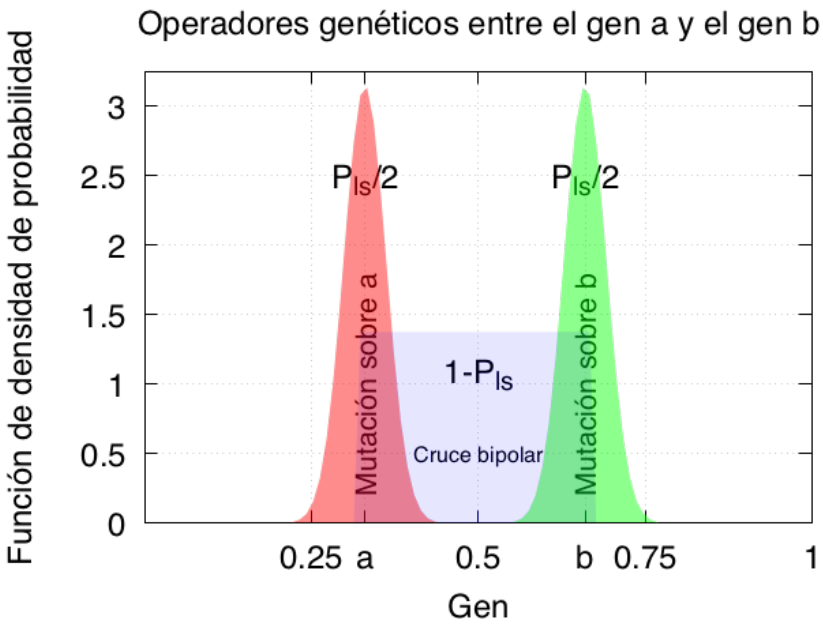


Figura 5.2: Representación de los operadores genéticos sobre dos genes

5.3.3. Reemplazo

En el algoritmo canónico se ha definido la probabilidad de un individuo de ser reemplazado en cada paso de tiempo calculado como muestra la ecuación 5.3. Esta probabilidad se obtiene a partir del tiempo de vida esperado, que representa un valor de tiempo de vida promedio para cada individuo según su calidad. Este tiempo de vida tiene que situarse entre T_{mat} (tiempo mínimo) y T_{max} (tiempo máximo). El tiempo de mínimo se denomina tiempo de maduración T_{mat}, y es un periodo en el que los individuos no se eliminan de la población ya que se considera un tiempo mínimo para poderse evaluar. El tiempo de vida máximo T_{max} representa el tiempo que se le permite a un individuo con la calidad óptima. La resta entre estos dos tiempos es lo que se denomina L (ecuación 5.4). La calidad de la que depende este tiempo de vida esperado T_{exp} es una calidad individual relativa a la calidad máxima conocida como se muestra en la ecuación 5.5. Finalmente, las fórmulas de T_{exp} son las que se muestran en las ecuaciones 5.6 y 5.7.

Para asignar esta esperanza de vida se ha definido una función en dos partes usando un modelo lineal. Las dos partes de la función están delimitadas por un valor de Q_r correspondiente a dos tercios de la calidad máxima $0{,}66Q_{max}$. La primera parte de la función son individuos que se consideran mediocres en calidad y la segunda parte de la función son individuos con mayor calidad (el tercio de individuos de mayor calidad). Si se observan las variables de esta ecuación, Q_i y Q_{max} son datos de la evolución y T_{max} es fijo, por lo tanto los parámetros intrínsecos que se pueden extraer de este proceso de reemplazo son c_m y T_mat. En la figura 5.3 se puede ver la representación gráfica de esta función fijando T_mat=10, c_m=0.2, T_{max}=100. En el eje x, se muestra la calidad Q_i del individuo, en el eje y izquierdo su tiempo de vida esperado, y en el eje y derecho la probabilidad de reemplazarse en ese paso de tiempo. Observando la gráfica, el parámetro c_m, que es el coeficiente de mediocridad, se muestra en la gráfica visualmente como un coeficiente que afecta al tiempo de vida esperada de un individuo (eje y izquierdo) con dos tercios de la calidad máxima, y que afecta a la pendiente de la recta que une este punto con el tiempo máximo de vida. Este coeficiente regula reemplazar con más o menos frecuencia de individuos de menor calidad. Si se opta por una estrategia que penaliza más la baja calidad, se puede fijar el parámetro c_m a un valor muy bajo, mientras que si se sigue una estrategia que penalice menos la baja calidad, se puede fijar el parámetro c_m a 1 que es su valor máximo. Hay que puntualizar que un parámetro c_m alto

implica menos reemplazos, pero aun así en este límite se garantiza un mínimo de reemplazos posibles.

$$P_{rep} = \frac{1}{T_{exp}} \tag{5.3}$$

$$L = T_{max} - T_{mat} \tag{5.4}$$

$$Q_r = \frac{Q_i}{Q_{max}} \tag{5.5}$$

$$T_{exp} = c_m Q_r \frac{3Q_{max}}{2} L \text{ if } Q_r \leq 0{,}66 Q_{max} \tag{5.6}$$

$$T_{exp} = (c_m + (Q_r - \frac{2Q_{max}}{3})(3 - 3cm))L \text{ if } Q_r > 0{,}66 Q_{max} \tag{5.7}$$

5.4. Pseudocódigo

Una vez descritos los parámetros intrínsecos del algoritmo canónico y los modelos probabilísticos a los que afectan, se presenta el pseudocódigo 3 que muestra el proceso del algoritmo canónico de DEE. Este código es el que se ejecutaría en cada uno de los robots. Dentro del bucle principal, cada robot obtiene su calidad y comprueba si debe entrar en modo de reproducción o en modo de reemplazo. Si le corresponde entrar en modo de reproducción, busca individuos en este modo, y si existen, se selecciona uno con la política de selección correspondiente. A continuación se genera un nuevo individuo que queda almacenado. Si corresponde entrar en el modo de reemplazo, este descendiente pasa a tomar el control de robot y se elimina al anterior controlador.

En el pseudocódigo 4 se muestra el proceso de recombinación y cómo se genera un nuevo genotipo. Primero hay que decidir cual de los dos padres va a ser la base del nuevo individuo, de forma aleatoria y con la misma probabilidad. A continuación, de manera aleatoria, en función del parámetro P_{ls}, se decide si se realiza una mutación o un cruce. La mutación es una ligera variación en una porción fija de los genes del genotipo. El cruce consiste en unir por un punto los

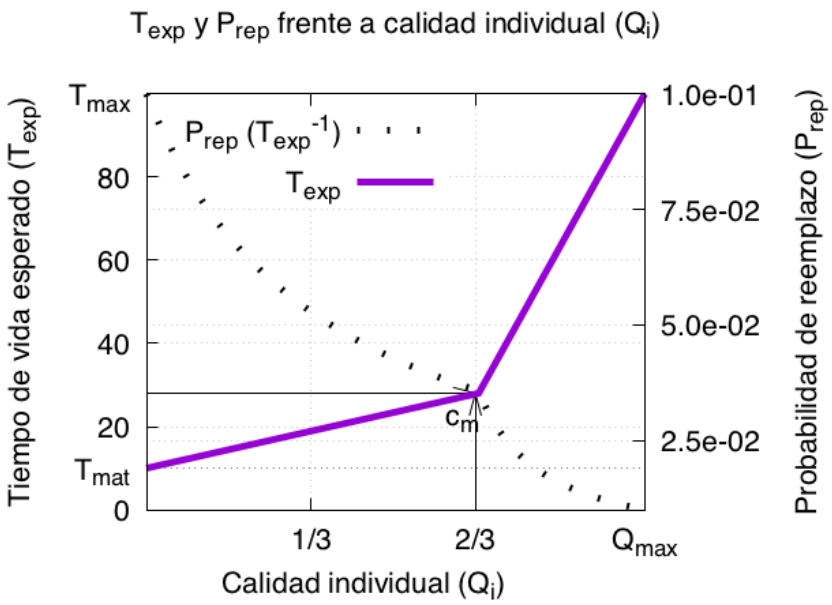


Figura 5.3: Representación gráfica de la función que parametriza el reemplazo, con $T_{mat} = 10$ y $T_{max} = 100$ y $c_m = 0{,}2$

dos genotipos, de forma que tiene una porción de cada uno de los dos genotipos de los padres.

Algorithm 3 Algoritmo canónico de dEE

```
for all robot r en robots do
  actuar en el entorno
  calcular calidad privada
  P_mating ← S_max / T_max {calcular probabilidad de reproducción}
  P_rep ← σ(f_t) {calcular probabilidad de reemplazo}
  if random < P_mating then
    buscar individuos para reproducción
    seleccionar individuo {Parámetro}
    aplicar operadores genéticos de recombinación {Parámetro}
  end if
  if tiempo de vida ≥ T_mat then
    if random() < P_rep or tiempo de vida ≥ T_max then
      reemplazar individuo por nuevo
      controlador[r] ← nuevoindividuo
    end if
  end if
end for
```

Para concluir este capítulo, se resumen a continuación los parámetros intrínsecos que definen al algoritmo canónico:

1. Tiempo de vida máximo T_{max}
2. Tamaño máximo de la ventana de selección S_{max}
3. Probabilidad de búsqueda local P_{ls}
4. Tiempo de madurez T_{mat}
5. Coeficiente de mediocridad c_m
6. Política de selección ψ_e

Una vez que se diseñado e implementado el algoritmo canónico DEE el siguiente paso para completar su formalización es analizar su funcionamiento básico, así como la sensibilidad y relevancia de los parámetros aquí establecidos, que es el objetivo del siguiente capítulo.

Algorithm 4 Recombinación

preferencia $\leftarrow$ 0,5
porcionMutacion $\leftarrow$ 0,15
porcionCruce $\leftarrow$ 0,15
if random $< preferencia$ **then**
  $\vec{\alpha} \leftarrow genotipoEmisor$
else
  $\vec{\alpha} \leftarrow genotipoReceptor$
end if
if random $< P_{ls}$ **then**
  for $i = 0$ to $length(\alpha)$ **do**
    if random $< porcionMutacion$ **then**
      $genotipoMutado_i \leftarrow \alpha_i + random * pasoMutacion$
    end if
  end for
  return genotipoMutado
else
  for $i = 0$ to $length(\alpha)$ **do**
    if random $< porcionCruce$ **then**
      $genotipoCruzado_i \leftarrow genotipoEmisor_i$
    else
      $genotipoCruzado_i \leftarrow genotipoReceptor_i$
    end if
  end for
  return genotipoCruzado
end if

Capítulo 6

Pruebas teóricas

En este capítulo, se describirán las pruebas teóricas realizadas para analizar formalmente el algoritmo canónico de dEE en diferentes escenarios. La metodología aplicada en esta tesis se puede dividir en tres etapas:

1. Análisis de los diferentes espacios de calidad que aborda un algoritmo de optimización distribuido en problemas reales. En la literatura de algoritmos evolutivos existen muchos estudios de espacios de calidad, debido a que el uso de estos algoritmos está mucho más extendido y a que es una parte esencial de su funcionamiento, pero dentro de problemas colectivos no se ha encontrado ningún estudio de este tipo. Por lo tanto, es necesario modelar los diferentes espacios de calidad que identifiquen los diferentes escenarios que podría abordar el algoritmo en un problema real

2. Definición de una serie de funciones benchmark. Estas funciones servirán para caracterizar de forma sintética cada uno de los diferentes espacios de calidad mencionados anteriormente y también para poder analizar la respuesta del algoritmo con estas funciones benchmark

3. Ejecución de un análisis de sensibilidad sobre las funciones: con el fin de comprobar la validez de la parametrización propuesta, se pretende ejecutar el algoritmo realizando un barrido en los rangos de los parámetros para analizar su comportamiento

6.1. Teoría sobre espacios de calidad

Para analizar los diferentes espacios de calidad en problemas colectivos, en primer lugar, hay que definir lo que es un espacio de calidad. Los espacios de calidad son una representación del conjunto de soluciones a un problema, y están formados por el espacio de búsqueda y la bondad o calidad de cada una de dichas soluciones a la hora de resolver dicho problema. Cada uno de los genotipos (soluciones candidatas) de la población se ha de evaluar en la función de calidad y se puede representar en un espacio, que es el espacio de calidad. Una forma sencilla de visualizar un espacio de calidad es en una representación de dos dimensiones, con genotipos de una única dimensión, en la que el eje x se representarían los diferentes genotipos de la población en un momento dado de la evolución y en el eje y se representarían las evaluaciones de calidad para cada uno de estos genotipos. En el caso de esta tesis, en la que la solución al problema es distribuida y la solución es toda la población, el espacio de calidad sería una evaluación global de toda la población, es decir del conjunto de todos los genotipos de todos los individuos, en cada paso de tiempo. Esta evaluación global representa el nivel de eficacia del equipo de robots realizando la tarea pero, a pesar de que este calidad global es lo que se maximiza, para que el algoritmo pueda operar de manera descentralizada, cada individuo tiene que conocer una calidad individual.

La asignación de calidad individual la debe establecer el diseñador para cada tarea cumpliendo tres condiciones. La primera es que la maximización de la calidad individual conlleve también la maximización de la calidad global (alineación de la calidad individual con la calidad global). La segunda condición es que permita establecer evaluaciones individuales de la contribución de cada genotipo a la calidad global, y por último, la tercera condición es que se pueda calcular de manera local y descentralizada. La alineación de la calidad individual con la global se ha abordado por diferentes autores, como por ejemplo Agogino y Tumer [Agogino and Tumer, 2008], con sus funciones de evaluación diferenciales, en las que las evaluaciones de un individuo, implican evaluar al equipo completo con ese individuo y posteriormente sin él, para calcular la diferencia de calidad y ser una medida de calidad individual alineada. En definitiva, esta alineación de la calidad es responsabilidad del diseñador.

Dentro de los espacios de calidad, distinguimos dos categorías en función de su dinamismo:

- los estáticos son aquellos en los que la población se mueve hacia el óptimo que siempre está situado en el mismo sitio

- los dinámicos son aquellos en los que el valor de los genotipos de la población influyen en la dirección en la que modificar la población para alcanzar el óptimo.

En el caso de esta tesis, debido a que está centrada en problemas colectivos, en el que la calidad de un individuo depende de la de otros individuos, los espacios de calidad son dinámicos. Del mismo modo, como la solución es todo el conjunto de genotipos, el espacio de calidad son evaluaciones de la calidad de todos los genotipos a lo largo del tiempo. Estas evaluaciones son de tipo global.

A continuación se tratarán y analizarán las definiciones de calidad global y privada y las casuísticas que estas generan en los problemas vamos a tratar.

6.1.1. Calidad global (Φ)

Se define por el objetivo global del problema que se quiere resolver y su asociación al genotipo está determinada por las reglas del entorno y de la codificación genotipo a fenotipo. Esta definición es independiente de la técnica evolutiva empleada. Considerando un entorno en el que solo un individuo resuelve la tarea, la calidad global se puede representar por una función de su genotipo γ y el tiempo t, como se puede ver en la ecuación 6.1. Sin perder generalidad, y para simplificar el análisis, el genotipo de n-dimensiones de un individuo se representará como una proyección en una sola dimensión.

$$\Phi(\gamma, t) \tag{6.1}$$

Cuando m individuos ($m > 1$) realizan la tarea y asumiendo una representación en una dimensión de sus genotipos, la calidad global se puede representar por un espacio genotípico de m dimensiones. La transformación de genotipo a fenotipo es la misma para todos los individuos y no se diferencian excepto en el genotipo, la calidad global $\Phi(\gamma_1, \gamma_2, ..., \gamma_n, t)$, con γ_i la i-ésima variable genotípica, tiene que cumplir la ecuación 6.2, que implica que la función de calidad es simétrica con respecto al hiperplano $\gamma_j = \gamma_i$.

$$\Phi(\gamma_1, \gamma_2, ..., \gamma_m, t) \equiv \Phi(P[\gamma_1, \gamma_2, ...\gamma_m], t) \text{ para cada permutación P} \quad (6.2)$$

Existe una vasta casuística de diferentes topologías en el espacio de calidad global, pero hay alguna características clave que son relevantes para el análisis de la respuesta de un algoritmo de dEE, porque son características típicas en los espacios de calidad de los problemas colectivos. Estas características son: la separabilidad genotípica y la combinación óptima de genotipos.

6.1.1.1. Separabilidad genotípica

El espacio de calidad global depende de los genotipos de los individuos y varía a lo largo de la evolución. Por lo tanto, su separabilidad está relacionada con la dependencia entre la contribución de un genotipo a la calidad global $[\delta\phi/\delta(\gamma_i)]$ y la contribución de los genotipos del resto de la población $[\delta(\delta\phi/(\delta\gamma_i))/(\delta\gamma_j)]$. Específicamente, si se considera que un genotipo define la respuesta de cada individuo, o aun mas, si define la especie a la que pertenece el individuo; la separabilidad genotípica de la función de calidad global puede ser vista como la interrelación entre especies del comportamiento colectivo resultante. Según esta interrelación, se pueden establecer tres tipos de espacios de calidad para cualquier función global: separable, pseudo-separable y no separable.

En un espacio separable, la contribución de cada individual no depende de los genotipos del resto de población. Formalmente, tomando como ejemplo una población de dos individuos con genotipos γ_1 y γ_2, un espacio separable es uno con una función de calidad global F como la que se muestra en la ecuación 6.3.

$$F = \phi(\gamma_1, \gamma_2, t) = \phi'(\gamma_1, t) + \phi'(\gamma_2, t) \quad (6.3)$$

$$\implies \frac{\delta(\frac{\delta\phi}{\delta\gamma_i})}{\delta\gamma_j} = 0; \forall(i,j)|i \neq j \quad (6.4)$$

Con ϕ' la contribución de cada individuo a ϕ. Una representación de un espacio de calidad separable de una población de dos individuos se muestra en la figura 6.1(a).

El segundo tipo de espacio es el pseudo separable, en donde la función de calidad F no puede ser definida directamente usando la contribución indepen-

diente de los individuos que forman parte de la población, pero al igual que en un espacio de calidad separable, las rutas de optimización individual no se ven afectadas por los genotipos del resto de la población. En este caso la fórmula se muestra en la ecuación 6.5

$$\frac{\delta sign(\frac{\delta F}{\delta \gamma_i})}{\delta \gamma_j} = 0; \forall (i,j) | i \neq j \tag{6.5}$$

Esta ecuación representa que las variaciones en el j-ésimo genotipo (γ_j), no afecta a la calidad del i-ésimo genotipo. Como consecuencia la ecuación

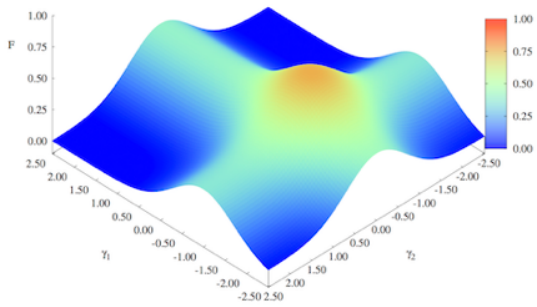
$$\frac{\delta(\gamma_i^{opt})}{\delta \gamma_j} = 0 \text{ con}$$
$$\gamma_i^{opt} = argmax_{\gamma_i} \phi(\gamma_1, \gamma_2, ..., \gamma_i, \gamma_m, t)$$

En la figura 6.1(b), se muestra una representación de un espacio pseudo separable.

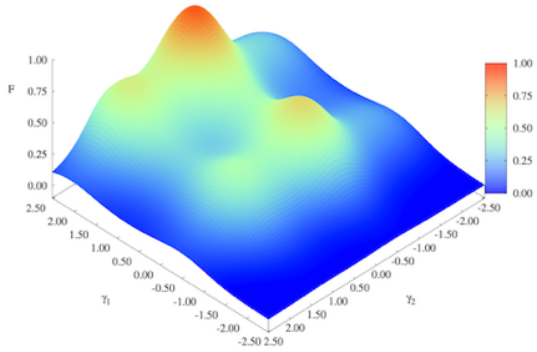
El tercer tipo de espacio es el espacio no separable e implica que la condición de pseudo separabilidad no se cumple en todo el espacio genotípico. Formalmente seguiría la ecuación 6.6, para al menos un subespacio del espacio genotípico m-dimensional γ^m. En la figura 6.1(c), en la parte inferior, se muestra una representación gráfica de este espacio considerando dos genotipos. En el este caso la optimalidad del genotipo de un individuo depende de los genotipos de los otros.

$$\frac{\delta sign(\frac{\delta F}{\delta \gamma_i})}{\delta \gamma_j} \neq 0 \tag{6.6}$$

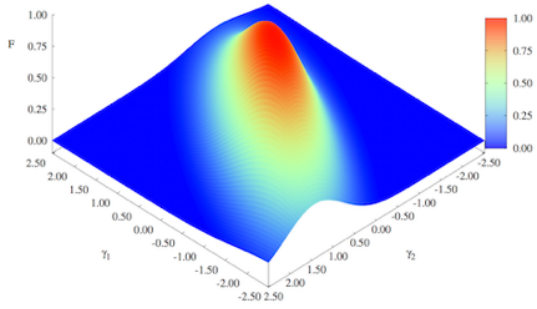
En términos de como afecta al proceso de optimización es importante resaltar que ajustar el comportamiento de un grupo de m individuos en un espacio separable o pseudo separable puede ser descompuesto en m procesos independientes de optimización (uno para cada individuo). Por otra parte, en un espacio no separable el problema no se puede descomponer y el algoritmo de optimización tiene que coordinar a todos los individuos teniendo en cuenta sus interdependencias. Los espacios no separables son muy relevantes en esta tesis y caracterizan problemas colectivos típicos.



(a) Espacio de calidad separable



(b) Espacio de calidad pseudoseparable



(c) Espacio de calidad no separable

Figura 6.1: Espacios de calidad considerados. (a) Espacio de calidad separable. (b) Espacio de calidad pseudoseparable. (c) Espacio de calidad no separable

6.1.1.2. Combinación óptima de genotipos

Otro aspecto de los espacios de calidad que es importante tiene relación con la configuración de la población una vez que se ha alcanzado el óptimo global, es decir, cual es la proporción de individuos en cada grupo cuando el óptimo global se ha alcanzado. El conjunto de genotipos óptimos constituye el objetivo del algoritmo de optimización y tiene dos aspectos: por una parte, la composición de la población óptima afecta al transcurso del proceso de optimización y por otra parte, representa el estado de equilibrio de un sistema dinámico oscilatorio. En este sentido, un aspecto muy relevante que debe ser considerado es el que trata con la variedad de genotipos presentes en el óptimo global. Independientemente de la separabilidad y de la modularidad del espacio de calidad, el óptimo puede ser alcanzado por la contribución colectiva de un conjunto de individuos y este conjunto óptimo puede ser categorizado en los siguientes casos:

- Espacio de una sola especie: el óptimo global se alcanza a través de un conjunto de genotipos idénticos:

$$\gamma_{1-sp}^{opt} \equiv \{\gamma_i^{opt}, \gamma_i^{opt}, \gamma_i^{opt}, ..., \gamma_i^{opt}\} \tag{6.7}$$

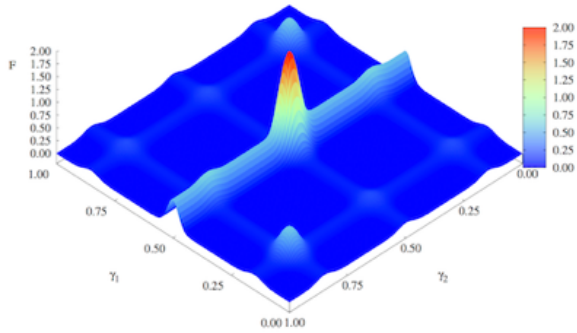
- Espacio multiespecies: el óptimo global corresponde a una combinación de dos o más genotipos diferentes:

$$\gamma_{m-sp}^{opt} \equiv \{\gamma_i^{opt}, \gamma_j^{opt}, \gamma_k^{opt}, \gamma_l^{opt}, \gamma_m^{opt}, ..., \gamma_z^{opt}\} \tag{6.8}$$

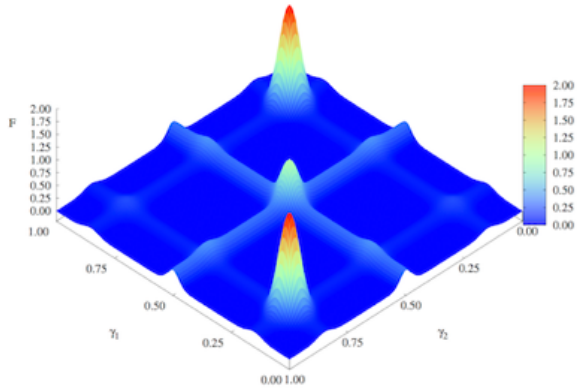
En términos prácticos, esta característica está relacionada con la especialización requerida por los componentes de la solución colectiva para resolver el objetivo global y corresponde al fenómeno biológico de la especialización. En la figura 6.2(a), se muestra una representación de un espacio con una especie y a su derecha, en la figura 6.2(b), una representación de un espacio de dos especies a la derecha.

6.1.2. Calidad privada (ψ)

La calidad privada representa el rendimiento asociado a cada individuo dependiendo de su comportamiento. Esta es la asignación de calidad que guía la evolución y no la calidad global. Esta distinción es muy importante en algo-



(a) Espacio de calidad de una especie



(b) Espacio de calidad de dos especies

Figura 6.2: Combinación óptima de genotipos. (a) Espacio de calidad con óptimo en una especie homogénea. (b) Espacio de calidad con óptimo en dos especies heterogéneas

ritmos de dEE porque la selección para reproducción y la esperanza de vida depende de esta calidad individual. Diseñar estas funciones no es fácil, especialmente en espacios globales no separables. Además, su definición establece las características del espacio de calidad privado, que es el optimizado para cada individuo. La calidad privada del individuo j-ésimo en la población (f_j) depende de su propio genotipo γ_j, los genotipos del resto de la población Γ_{-j} y de las dinámicas del entorno (t).

$$f_j = \psi(\gamma_j, \Gamma_{-j}, t) \text{ con } \Gamma_{-j} \equiv \{\gamma_1, \gamma_2, \gamma_3, ...\gamma_{j-1}, \gamma_{j+1}, \gamma_n\} \tag{6.9}$$

Considerando una población de dos individuos, su función de calidad privada f puede ser formulada como:

$$f_1 = \psi(\gamma_1, \gamma_2, t), f_2 = \psi(\gamma_1, \gamma_2, t) \tag{6.10}$$

La primera característica a estudiar de los espacios de calidad privada, y también una de los más relevantes e interesantes que afectan al rendimiento de un algoritmo dEE, es la dependencia de la calidad privada de un individuo a los calidades privadas del resto de la población. Hay similitudes entre esta característica y la separabilidad usada para la casuística de la calidad global, pero son conceptos diferentes. La calidad global es siempre multidimensional por definición y esta dimensionalidad permite a las variables ser separables o no. En el caso de la calidad privada, el espacio de calidad asignado a cada individuo es estrictamente unidimensional en términos del proceso de búsqueda, pero como este espacio de una dimensión puede variar debido a cambios de los genotipos del resto de individuos, estos genotipos son considerados dimensiones adicionales. Por lo tanto, se puede diferenciar dos tipos de espacios de calidad privada:

- espacios unidimensionales: la calidad privada no depende de los genotipos del resto de la población. Si la función de calidad global se puede expresar como la suma de funciones de calidad privadas unidimensionales, el espacio de calidad global será necesariamente separable.

$$\frac{\delta f_i}{\delta \gamma_j} = 0 \ \forall j \neq i \tag{6.11}$$

- Espacios multidimensionales: para cada subespacio del espacio genotípico,

la calidad privada debe cumplir:

$$\frac{\delta f_i}{\delta \gamma_j} \neq 0 \text{ con j} \neq i \tag{6.12}$$

que es el resultado de la creación de una calidad privada para el caso de un espacio de calidad global no separable.

Dentro de este caso de espacio multidimensional, pero en un nivel de interacción entre especies, es interesante estudiar la interdependencia entre la calidad de los individuos de la población, que puede ser definida usando la función I_f:

$$I_f^{i,j}(\Gamma) = \Delta_j f(\Gamma) + \Delta_i f(\Gamma) \tag{6.13}$$

$$\text{con } \Delta_k f = \frac{\delta f_i}{\delta \gamma_k} \frac{\delta f_j}{\delta \gamma_k} \tag{6.14}$$

Basado en esta interdependencia, se puede diferenciar una casuística que es muy relevante desde el punto de vista práctico:

- Subespacio colaborativo ($I_f^{i,j}(\Gamma) > 0$): está representado por un subespacio en donde las variaciones genotípicas producen un incremento de calidad en ambos individuos (i-ésimo y j-ésimo). Un ejemplo de este tipo de espacio para un sistema de dos individuos se muestra en la figura 6.1.2, donde está claro como la calidad máxima se alcanza con el mismo genotipo para los dos individuos
- Subespacio competitivo ($I_f^{i,j}(\Gamma) < 0$): está representado por un subespacio en donde las variaciones genotípicas producen efectos contrarios en la calidad de los individuos. Si los comportamientos óptimos de los individuos están contenidos en un subespacio competitivo, las áreas de calidad máxima no pueden coincidir. En este caso las configuraciones optimas para los individuos son diferentes y se producirá una competición entre los individuos para maximizar su propia calidad cambiando las configuraciones en su propio beneficio. Un ejemplo de este tipo de espacio para un sistema de dos individuos se muestra en el medio de la figura 6.1.2, donde se observa que los valores genotípicos que maximizan una función y los del otro son contrarios.
- Subespacio neutro ($I_f^{i,j}(\Gamma) \approx 0$): hay dos opciones en este caso; o el efecto de un genotipo sobre otro es muy pequeño o un genotipo pro-

duce colaboración y otro competición. En cualquiera de estos casos, si esta condición se cumple, el proceso de optimización será guiado por el genotipo que produce o neutralidad o colaboración.

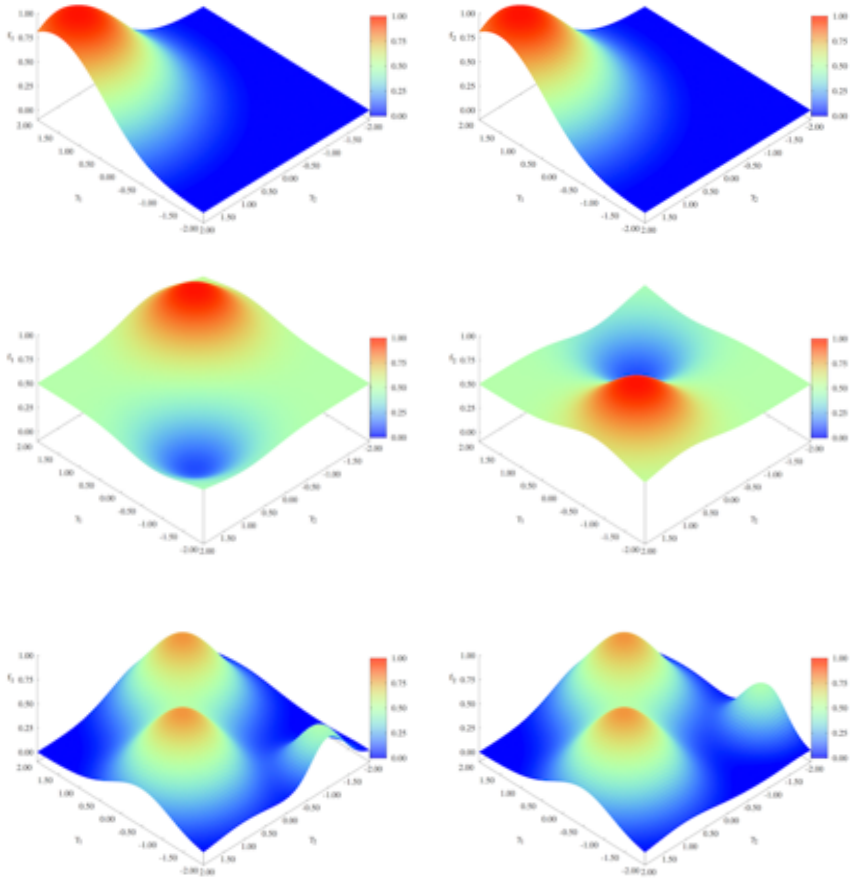


Figura 6.3: Espacios de calidad privada para dos individuos de una población. La fila de arriba corresponde a espacios colaborativos, la del medio a espacios competitivos y la de abajo a espacios neutrales

6.2. Espacios de calidad seleccionados

El análisis realizado en esta tesis considera funciones de calidad global aditivas para definir una medida de calidad en cada caso. Es decir, la calidad global se calcula como la suma de calidades individuales de todos los individuos en la población. Consecuentemente, la definición de la calidad individual debe considerar que la suma de cada contribución individual debe llevar a la maximización de la calidad global. La calidad individual se ha normalizado en el rango [0:1] para que la calidad global se sitúe desde 0 hasta el tamaño de la población. La calidad asociada a cada individuo es calculada usando una formula matemática que relaciona el genotipo a un valor de calidad individual, como si se considerara una codificación directa. Tal y como se había mencionado previamente, la evolución está guiada por la calidad individual, pero la casuística depende de la calidad global. Por lo tanto, se han definido diferentes funciones de calidad global y las funciones de calidad individual se han calculado en base a ellas. Con el objetivo de cubrir todo el abanico de espacios de calidad, se han definido cuatro funciones de calidad globales:

1. Separable, unimodal, calidad global con óptimo de una especie: existen varios problemas colectivos en optimización que requieren que todos los individuos posean el mismo genotipo, es decir, que realicen la misma tarea. Son problemas colectivos pero en una tarea no colaborativa. Por ejemplo, en un problema de limpieza colectiva o un problema de rutas de vehículos, todos los individuos deben tener el mismo comportamiento para obtener la calidad global más alta. El espacio de calidad es claramente separable porque cada agente (robot o vehículo) obtiene la recompensa más alta independientemente del resto de agentes. A partir de esta función de calidad global, se ha definido una función individual de una dimensión con un único óptimo que se muestra en la ecuación 6.15.

$$f_j(\gamma_j, t) = 0{,}9e^{-\alpha dif(\gamma_j - 0{,}5)^2} + 0{,}1e^{-\alpha 0{,}1 dif(\gamma_j - 0{,}5)^2} \tag{6.15}$$

   Como se muestra en la figura 1, la función está basada en una función Gaussiana donde el genotipo óptimo es 0.5 con una calidad máxima de 1. Cualquier otro valor del genotipo obtiene peor calidad. Para analizar la respuesta del algoritmo canónico a diferentes complejidades del espacio de búsqueda, se ha introducido un coeficiente de dificultad. Esta dificultad

establece la anchura de la campana de Gauss, haciéndola más estrecha a medida que aumenta la dificultad, como se muestra en la figura 1. El nivel de dificultad se ha variado de 1 hasta 10. La dimensionalidad del cromosoma también se ha variado con el objetivo de analizar la respuesta del algoritmo en espacios de calidad con más dimensionalidad, como los que surgen en problemas reales. La fórmula que generaliza para cualquier dimensionalidad se muestra en la ecuación 6.16

$$f_j(\gamma_j, t) = 0{,}9e^{-\alpha difR(\sum_d(\gamma_{j,d}-0{,}5)^2} + 0{,}1e^{-\alpha 0{,}1difR\sum_d(\gamma_{j,d}-0{,}5)^2} \quad (6.16)$$

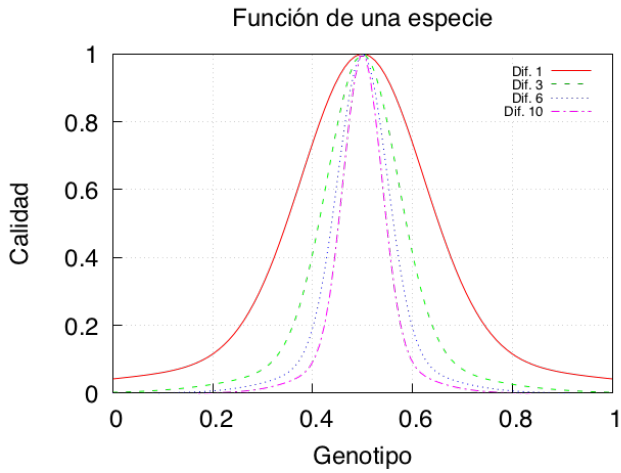


Figura 6.4: Función 1: separable, unimodal, óptimo de una especie

2. No separable, multimodal, calidad global con óptimo de dos especies: muchos de los problemas colectivos requieren especialización, es decir, individuos con diferentes genotipos asociados a diferentes comportamientos deben cooperar conjuntamente. El caso más simple en un problema colectivo con dos óptimos donde los individuos se deben distribuir entre ellos, dando lugar a un espacio de calidad no separable con dos especies. Es decir, la calidad global define un problema donde la solución óptima se alcanza cuando un número específico de individuos tienen un determinado genotipo y el resto de individuos tiene otro determinado genotipo.

Corresponde a una tarea colectiva y colaborativa. Por ejemplo, en una tarea de limpieza colectiva, la calidad global corresponde a una configuración donde se limpia mejor cuando una parte de los individuos barren el suelo y otra parte de los individuos lo friegan. La no separabilidad se incluye al establecer una dependencia entre las dos tareas. En el ejemplo mencionado, significa que el numero de individuos realizando la tarea de barrer afecta la recompensa por la tarea de fregar y viceversa. La función de calidad individual que supone la maximización de la función de calidad global tiene un valor óptimo variable porque depende de los genotipos del resto de los individuos (de acuerdo con la no separabilidad de la calidad global). Esto implica que existe una distribución óptima de individuos de las dos especies que deben ser encontradas por el algoritmo. La función de calidad individual, para el caso unidimensional, que ha sido aplicada para este tipo de espacio se muestra en la ecuación 6.17.

$$f(\gamma_j, \Gamma_{-j}, t) = \begin{cases} 0{,}9C_1(\Gamma_{-j})e^{-\alpha dif x_{0,15}^2} + \\ 0{,}1C_1(\Gamma_{-j})e^{-\alpha 0{,}1 dif x_{0,85}^2} & \text{si } x_{0,15} \equiv min(x_i) \\ 0{,}9C_2(\Gamma_{-j})e^{-\alpha dif x_{0,85}^2} + \\ 0{,}1C_2(\Gamma_{-j})e^{-\alpha 0{,}1 dif x_{0,85}^2} & \text{si } x_{0,85} \equiv min(x_i) \end{cases} \tag{6.17}$$

$$\text{con } x_a^2 = R\sum_d (\gamma_{j,d} - a)^2 \text{ , i} = \{0{,}15, 0{,}85\}$$

$$C_1 = min(\frac{0{,}3 + 0{,}7\sum_1^n e^{-\alpha dif x_{0,85}^2}}{0{,}5 \times populationSize}, 1) \tag{6.18}$$

$$C_2 = min(\frac{0{,}3 + 0{,}7\sum_1^n e^{-\alpha dif x_{0,15}^2}}{0{,}5 \times populationSize}, 1) \tag{6.19}$$

Como se puede ver en la figura 6.5, la función Gaussiana en este caso se localiza en los valores del genotipo 0.15 y 0.85, pero con la calidad máxima dependiendo ahora de la distribución de genotipos en la población, como puede verse en la gráfica de la derecha de esta figura. Para esta tesis, se ha considerado el caso en el que la calidad máxima se obtiene con una distribución perfectamente balanceada, es decir, con mitad de la población con el genotipo 0.15 y la otra mitad con el genotipo 0.85, como se muestra en la parte izquierda de la figura 6.5. Para formalizar este con-

cepto, la ecuación anterior tiene dos coeficientes C_1 y C_2, que representan el porcentaje de individuos en cada óptimo, C_1 en la campana en 0.85 y C_2 en la campana en 0.15. Las fórmulas para calcular estos coeficientes se muestran en las ecuaciones 6.18 y 6.19. Cuando la distribución de la población está perfectamente balanceada, $C_1 = 1$ y $C_2 = 1$, se obtiene una calidad máxima, pero en cualquier otra distribución por ejemplo, con un 25 % de los individuos en la campana de 0.15, C_1 tendría un valor de 0.75 y C_2 un valor de 0.25, ya no se obtiene la calidad máxima. En este caso el algoritmo debería balancear la población hacia la campana con menos individuos, de forma que cuando lo haya conseguido, todos los genotipos obtendrán la calidad máxima. Explicado de otro modo, los genotipos situados en los valores 0.15 y 0.85 que son los óptimos, no obtendrán calidad individual máxima, hasta que la distribución este balanceada, la mitad de ellos en un valor y la otra mitad en otra.

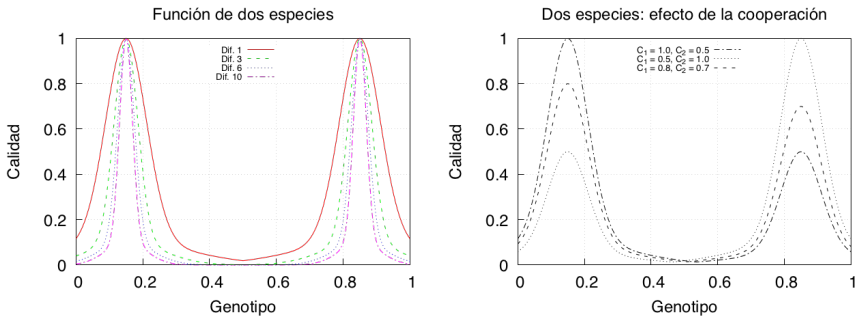


Figura 6.5: Función 2: óptimo en dos especies heterogéneas

3. Pseudoseparable, multimodal, calidad global multiespecie: esta configuración en la que existen dos óptimos globales no separables y de dos especies, pero en donde adicionalmente existe un óptimo local correspondiente a un único óptimo separable, es un caso muy frecuente que merece ser incluido en el análisis. Este caso describe un problema en el que un balance entre dos especies ofrece la mejor solución pero en donde una solución con una especie es posible también, aunque con un nivel de calidad menor. Por ejemplo, podría corresponder a un problema de limpieza en donde una distribución balanceada entre robots que barren y robots que friegan soluciona el problema de forma óptima, pero una configuración con robots

que realizan ambas tareas también solucionaría el problema aunque con menor eficiencia. En este caso, la calidad individual tiene tres óptimos (uno de ellos subóptimo). La fórmula para esta configuración se muestra en la ecuación 6.20 definida en tres partes, según si el valor del genotipo esta más próximo a 0.15, a 0.5 o a 0.85. En la figura 6.6, se puede observar la gráfica para esta configuración.

$$f(\gamma_j, \Gamma_{-j}, t) = \begin{cases} 0{,}9C_1(\Gamma_{-j})e^{-\alpha dif x_{0,15}^2} + & \\ 0{,}1C_1(\Gamma_{-j})e^{-\alpha 0{,}1 dif x_{0,15}^2} & \text{si } x_{0,15} \equiv min(x_i) \\ 0{,}54e^{-\alpha dif x_{0,5}^2} + & \\ 0{,}06e^{-\alpha 0{,}1 dif x_{0,5}^2} & \text{si } x_{0,5} \equiv min(x_i) \\ 0{,}9C_2(\Gamma_{-j})e^{-\alpha dif x_{0,85}^2} + & \\ 0{,}1C_2(\Gamma_{-j})e^{-\alpha 0{,}1 dif x_{0,85}^2} & \text{si } x_{0,85} \equiv min(x_i) \end{cases} \tag{6.20}$$

$$\text{con } x_a^2 = R\sum_d (\gamma_{j,d} - a)^2 \text{ , i} = \{0{,}15, 0{,}5, 0{,}85\}$$

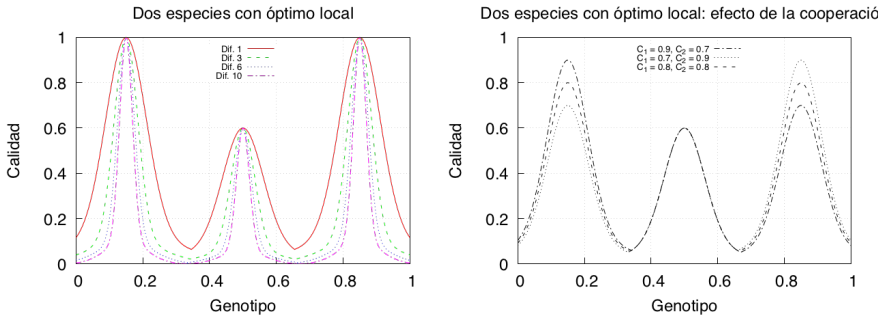


Figura 6.6: Función 3: óptimo con dos especies heterogéneas pero con óptimo local en genotipo homogéneo

4. Pseudoseparable, unimodal, calidad multiespecie: una configuración similar a la anterior pero en la que la solución óptima se sitúa en el único óptimo separable, es decir, existen dos óptimos locales que corresponden a dos especies balanceadas pero proveen una solución de menor calidad que la solución con el óptimo local. Siguiendo el ejemplo anterior de limpieza colectiva, los robots que realizan las dos tareas tendrían mejor rendimien-

to que una solución con la población balanceada entre robots que realizan cada una de las dos tareas de forma especializada. La función de calidad individual tiene tres óptimos con dos óptimos locales y se muestra en la ecuación 6.21. La gráfica de la función se muestra en la figura 6.7 y como se ve, el balance de especies en las dos zonas correspondientes a los valores 0.15 y 0.85, no afectan a la calidad de la zona central, que es la óptima.

$$f(\gamma_j, \Gamma_{-j}, t) = \begin{cases} 0{,}3C_1(\Gamma_{-j})e^{-\alpha dif x_{0,15}^2} + & \\ 0{,}03C_1(\Gamma_{-j})e^{-\alpha 0{,}1 dif x_{0,15}^2} & \text{si } x_{0,15} \equiv min(x_i) \\ 0{,}81e^{-\alpha dif x_{0,5}^2} + & \\ 0{,}09e^{-\alpha 0{,}1 dif x_{0,5}^2} & \text{si } x_{0,50} \equiv min(x_i) \\ 0{,}3C_2(\Gamma_{-j})e^{-\alpha dif x_{0,85}^2} + & \\ 0{,}03(\Gamma_{-j})e^{-\alpha 0{,}1 dif x_{0,85}^2} & \text{si } x_{0,85} \equiv min(x_i) \end{cases} \tag{6.21}$$

$$\text{con } x_a^2 = R\sum_d (\gamma_{j,d} - a)^2 \text{ , i} = \{0{,}15, 0{,}5, 0{,}85\}$$

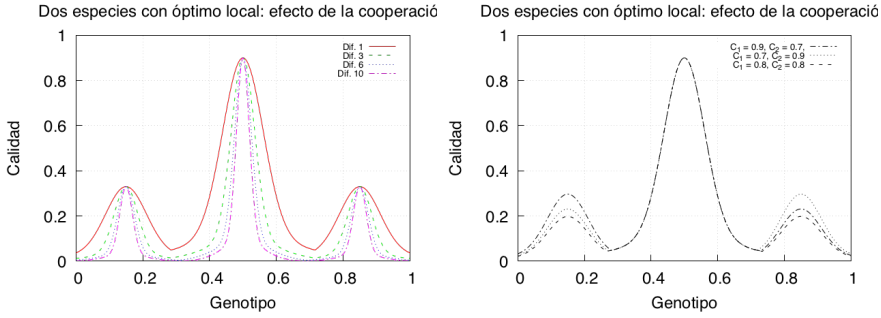


Figura 6.7: Función 4: función de dos especies con óptimo global en genotipo homogéneo

6.3. Análisis de sensibilidad

6.3.1. Configuración experimental

Una vez definidas las funciones de calidad, se presentará aquí el resto de la configuración del experimento para realizar el análisis de sensibilidad del algoritmo canónico de dEE. Con respecto los parámetros del algoritmo, se han considerado los siguientes parámetros.

- Codificación genética: el genotipo se codifica como un array de valores reales de una dimensión fija. En este capítulo se han probado diferentes codificaciones con diferentes dimensionalidades. Cada gen está normalizado entre en el intervalo [0,1].
- Reproducción
  - Selección para reproducción: el valor de tiempo de vida máxima T_{mat} se ha fijado a 10^3 en todos los experimentos. Por lo tanto, lo probabilidad la reproducción se define simplemente por el tamaño máximo de la ventana de selección S_{max}, que será uno de los parámetros estudiados en este análisis de sensibilidad.
  - Política de selección: teniendo en cuenta la definición de la calidad usada para estudiar el comportamiento del algoritmo, solo se puede tener en cuenta un criterio basado en calidad o basado en genotipos; para este análisis se usará un criterio basado en calidad.
  - Recombinación genotípica: los operadores seleccionados para utilizarse en este estudio son un operador de mutación y otro de cruce bipolar, como el que se describen en el capítulo que describe el algoritmo canónico.
- Reemplazo
  - Tiempo de madurez: este parámetro afecta de dos modos a la operación del algoritmo. Por una parte afecta al número de iteraciones requeridas para evaluar a un individuo de una forma fiable y por otra parte afecta al periodo de tiempo que se le asigna a un individuo con calidad privada igual a cero, para permitir cierta diversidad en la población. El tiempo de madurez afecta en estos dos niveles y

por lo tanto el valor óptimo para este parámetro es el que permita al algoritmo evaluar a un individuo de forma fiable y que asigne el tiempo de vida correcto a individuos muy ineficientes.

- Calidad máxima actual: para ajustar este valor durante la evolución de forma descentralizada, cada individuo almacena un valor de calidad máxima actual y cada vez que tiene una recombinación con otro individuo se transmite. Se si recibe un valor mayor, se actualiza. Debido a que cambia durante la evolución y se propaga por la evolución este parámetro no será incluido en el análisis de sensibilidad.
- Coeficiente de mediocridad: este coeficiente será estudiado en el análisis sin ninguna simplificación.

Resumiendo, se ha definido un experimento con el objetivo de analizar como la parametrización del algoritmo canónico afecta a su rendimiento en problemas típicos de optimización distribuida. Las siguientes funciones de calidad y parámetros a considerar son:

- Funciones de calidad:
  1. Función 1: una especie, separable, (genotipos homogéneos óptimos, tarea no colaborativa)
  2. Función 2: dos especies, no separable, (genotipos heterogéneos óptimo, tarea colaborativa óptima)
  3. Función 3: multi especie, pseudo separable, (genotipos heterogéneos óptimos, tarea colaborativa óptima con óptimos locales)
  4. Función 4: multi especie, pseudo separable, (genotipos homogéneos óptimos, tarea óptima no colaborativa)
- Parámetros:
  1. Tamaño máximo de la ventana de selección S_{max}
  2. Probabilidad de búsqueda local P_{ls}
  3. Tiempo de madurez T_{mat}
  4. Coeficiente de mediocridad c_m

Para aplicar el algoritmo cDEE sobre este conjunto de funciones se va a utilizar del pseudocódigo 5, que parte del establecido en el capítulo 5, pero donde

se han definido operadores concretos a la hora de realizar su implementación práctica. Así, la recombinación genotípica se realizará en base a los operadores de mutación y cruce que usaba el algoritmo ASiCo, y que se explicaron en el capítulo 5.2. De este algoritmo también se seguirá el concepto de embrión, y por lo tanto, los nuevos genotipos generados se almacenarán en cada robot en un embrión hasta que se produzca el reemplazo de los individuos que los portan.

Algorithm 5 Algoritmo cDEE

```
for all robot r en robots do
  calcular calidad privada
  P_mating ← S_max / T_max {calcular probabilidad de reproducción}
  P_rep ← σ(f_t) {calcular probabilidad de reemplazo}
  if random < P_mating then
    buscar individuos para reproducción
    seleccionar individuo {Parámetro}
    aplicar mutación o cruce {Parámetro}
  end if
  if tiempo de vida ≥ T_mat then
    if random() < P_rep or tiempo de vida ≥ T_max then
      reemplazar con el embrión
      controlador[r] ← siguienteControlador
    end if
  end if
end for
```

La primera dificultad que surge al realizar un análisis de sensibilidad sin ningún conocimiento previo de los parámetros óptimos es el de la escala de barrido de los parámetros. Dependiendo en la granularidad del parámetro que esta siendo barrido puede implicar un gran numero de combinaciones de parámetros, requiriendo un coste computacional muy grande, o un numero menor de combinaciones con una probable falta de información. Además, cuando se realiza un análisis de sensibilidad es interesante estudiar no solo el comportamiento del algoritmo con una parametrización concreta, sino también su estabilidad y cómo de repetibles son los resultados para un conjunto de parámetros. El hecho de tener una respuesta muy variable sería una debilidad del algoritmo evolutivo que no lo haría aplicable a ciertos problemas debido a esta inestabilidad. Con el objetivo de evitar un experimento de prueba y error, se ha realizado un pre-análisis del espacio parámetros-calidad para determinar un rango más estrecho para los parámetros antes de realizar el estudio exhaustivo. Este pre-análisis está basado en el método de innovization y se describe en la siguiente sección.

6.3.2. Preanálisis con el método de innovization

Deb et. al. presentaron el método de innovization en [Deb and Srinivasan, 2006] para obtener principios de diseño innovadores sobre variables de decision en un problema de optimización. Está basado en la optimización multi objetivo y el análisis de las soluciones óptimas. Concretamente, este método obtiene el frente de Pareto para un problema multi objetivo con el algoritmo NSGA-II y posteriormente se analiza la solución para descubrir tendencias y compromisos entre las soluciones. En esta sección, y mediante el uso de esta técnica, se analizará el dominio de los cuatro parámetros intrínsecos con los principios de innovization, con el objetivo de obtener más información sobre los rangos relevantes de los parámetros para las diferentes funciones de calidad consideradas. El algoritmo NSGA-II empleado aquí tiene la siguiente configuración:

- El cromosoma esta formada por 4 genes, cada uno correspondiendo a un parámetro intrínseco a optimizar. Aunque los genes esta normalizados entre [0:1] en evolución, su rango de valores reales es el siguiente:
  - Tamaño máximo de ventana de selección (S_{max}): [1:Tamaño población]
  - Probabilidad de búsqueda local (P_{ls}): [0:1]. Un valor de 1 corresponde a solo búsqueda local (mutación) mientras que un valor de 0 corresponde a solo exploración (cruce)
  - Tiempo de madurez (T_{mat}): [1:10^3]. Varia desde el numero mínimo de iteraciones que un individuo necesita para ser evaluado al tiempo máximo de vida (T_{max}) establecido.
  - Coeficiente de mediocridad (C_m): [0:1]. Considerando el tiempo mínimo de vida (T_{mat}) y el tiempo máximo T_{max}, el tiempo de vida de un individuo mediocre con $c_m = 0$ corresponde a T_{mat} y con c_m=1 a T_{max}
- Hay dos objetivos en la evolución desarrollada por NSGA-II: la maximización de la calidad global y la minimización de la variabilidad de la calidad global. Estos dos valores se miden probando cada configuración varias veces y calculando la media y la desviación estándar de la calidad global obtenida
- La función de calidad son las cuatro funciones establecidas en la sección anterior. En particular cada individuo en el NSGA-II se ejecuta 25 veces

Tabla 6.1: Parametros específicos para el NSGA-II

Parámetro	Valor
Tamaño población	50
Generaciones	100
Selección	Torneo
Tamaño pool	2
SBXCrossover	5 %
PolynomialMutation	65 %

durante 10^4 iteraciones con los valores específicos de los 4 parámetros intrínsecos incluidos en el correspondiente cromosoma. La calidad final de cada ejecución es el valor medio de la calidad global para tener en cuenta no solo la calidad al final de la evolución sino la tendencia, teniendo en cuenta que los algoritmos dEE evolucionan con final abierto. Para cada función, se han considerado dos niveles de dificultad (1 y 5) y dos dimensionalidades diferentes (2 y 5)

- Los parámetros específicos del algoritmo NSGA-II se muestran en la siguiente tabla.

6.3.2.1. Análisis de la población

El tamaño de la población en un algoritmo evolutivo es un parámetro muy relevante para su rendimiento, ya que tiene una relación directa con la variabilidad de las soluciones candidatas. Además, los algoritmos de dEE son incluso más sensibles a este parámetro debido a su naturaleza distribuida. Pero hay una desventaja al utilizar poblaciones muy grandes, que es el coste computacional. Para el mismo coste computacional, cuanto mayor la población, menor numero de generaciones que el sistema puede evolucionar. Por lo tanto, hay que encontrar un compromiso. En el caso del algoritmo dEE y cuando la aplicación final es un entorno real, no se tendría esta limitación en coste computacional pero si otras limitaciones en otros recursos, por ejemplo, un presupuesto limitado para comprar robots, que limitaría el número de individuos. Afortunadamente, se verá que a partir de un tamaño de población, el rendimiento del algoritmo no mejora significativamente. Con el objetivo de no definir este tamaño de población arbitrariamente, se ha realizado un análisis con el método de innovization

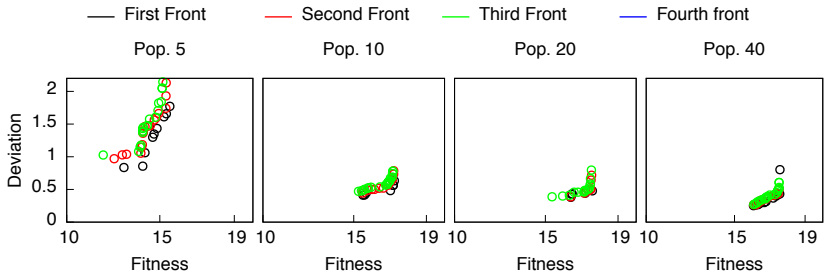


Figura 6.8: Frentes de Pareto para diferentes tamaños de población en el método de innovization para la función 2

para cuatro tamaños de población: 5, 10, 20 y 40. En este análisis solo se ha usado la segunda función de calidad (dos especies, no separable), ya que es la más representativa, en dos dimensiones, con el resto de parámetros como se ha descrito anteriormente.

Los resultados se muestran en la figura 6.8. De izquierda a derecha las gráficas corresponden a tamaños de población desde 5 hasta 40. Las gráficas representan tantos frentes de Pareto como se requieran para contener las 30 mejores soluciones en términos de dominancia. Cada frente se muestra con un color diferente, siendo los cuatro primeros frentes: negro, rojo, verde y azul respectivamente. Aunque los rangos reales de la calidad global van de [0:tamaño de la población], se han normalizado a [0:20] en todos los experimentos para poder comparar resultados fácilmente. Los frentes de Pareto muestran mucha información y varios aspectos pueden ser analizados cuando se comparan diferentes pruebas. En primer lugar, la calidad máxima obtenida independientemente de la desviación, mejora significativamente de 5 individuos a 10. Hay también una mejora en calidad global desde 10 a 20, pero no de 20 a 40. En cuanto a desviación, se reduce mucho al pasar de 5 a 10 pero no se reduce significativamente a partir de 10. A partir de estos resultados, se puede considerar que una población de 20 es un buen compromiso entre calidad, desviación y coste computacional, y por eso es el tamaño de población que se ha seleccionado para el resto de análisis de sensibilidad.

6.3.2.2. Función de calidad de una especie, separable (tarea no colaborativa)

La primera función de calidad global a analizar es la función de una especie y separable (función 1). Los resultados se muestran en la figura 6.3.2.2. Las cuatro gráficas de la parte superior muestran los frentes de Pareto para mostrar al menos 30 puntos, para las cuatro combinaciones de dimensionalidad y dificultad establecidas anteriormente. Como se puede observar, para todas las pruebas existen configuraciones que son capaces de alcanzar una calidad mayor que 19, lo que significa que alcanzan el óptimo. Para esta función, esto requiere que todos los genotipos sean prácticamente idénticos. Se puede observar el efecto del incremento de dificultad que produce un ligero decremento de calidad y un aumento considerable de la desviación.

Las cuatro gráficas de la parte inferior corresponden a los valores específicos de los parámetros c_m, T_{mat}, P_{ls} y S_{max} para los mismos frentes de Pareto mostrando el valor de calidad media que producen. En este caso se muestran de diferente color según la dificultad y dimensionalidad. De izquierda a derecha, en la gráfica de c_m, se puede observar que todos se encuentran por debajo del valor 0.4 y la mayoría cercanos a cero, lo que supone que el NSGA-II no pudo encontrar configuraciones con buena calidad y desviación cuando c_m es mayor que 0.4. En términos de comportamiento del algoritmo, esto implica que los individuos con calidad individual baja, se eliminan de la población rápidamente. Continuando con el tiempo de madurez T_{mat}, se puede observar una línea horizontal en este parámetro con valor 1 ($log_{10} = 0$), que significa que los individuos tienen un tiempo de vida mínimo igual a 1. La probabilidad de búsqueda local P_{ls} siempre es menor que 0.5, que significa que la mejor estrategia para resolver esta espacio de calidad es una estrategia muy explorativa, con más cruce que mutación. Esto sucede porque en espacios de calidad simples, el operador de cruce produce soluciones que convergen de forma rápida sin los problemas que le ocurren a este operador en espacios de calidad más complejos, por ejemplo en espacios de dos especies. Por último, con respecto al parámetro S_{max}, se puede ver que existen valores en todo el rango pero hay tendencia de que los valores de calidad altos corresponden a valores de S_{max} altos. Esto es un resultado esperado ya que aumentar S_{max} aumenta la variabilidad de los candidatos para reproducción y esto favorece al algoritmo.

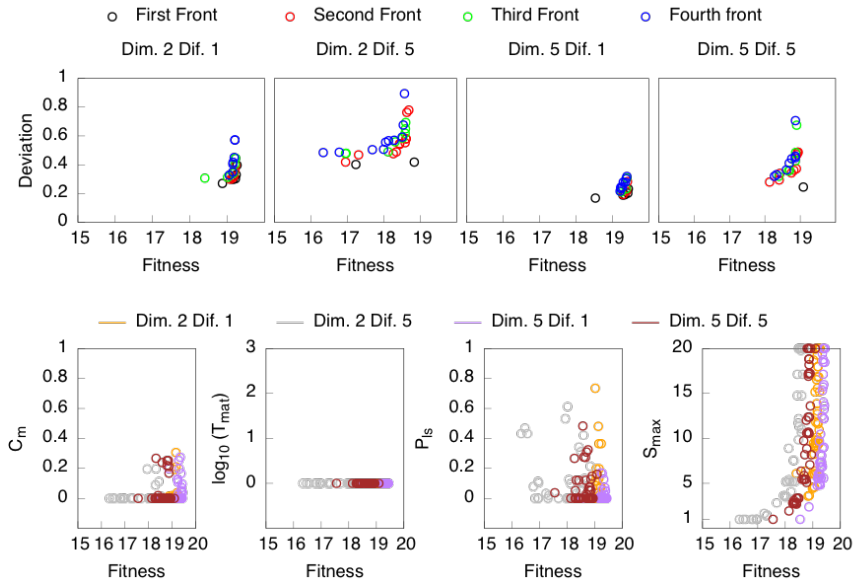


Figura 6.9: Resultados del método de innovization para la función 1. En la fila de arriba se muestran los frentes de Pareto para contener al menos 30 puntos. En la fila de abajo se muestra el comportamiento de los parámetros intrínsecos.

6.3.2.3. Función de dos especies, no separable (tarea colaborativa)

En la figura 6.3.2.3 se muestran los resultados de aplicar el método de innovization sobre la función global de dos especies y no separable (función 2). El formato de la figura es exactamente el mismo que para la anterior función. Lo primero que se puede comprobar es que el problema es más complejo que para el caso anterior ya que el valor de calidad media máxima que alcanza el algoritmo es menor, cercano a 18, que el valor que se alcanzaba en la función anterior, cercano a 19. Hay que tener en cuenta para esta función, que el valor cercano a 17 solo se puede alcanzar si la mayoría de la población está correctamente balanceada entre los dos óptimos, así que se puede concluir que el algoritmo obtiene las dos especies requeridas. De misma forma que para la función anterior, el nivel de dificultad decrementa levemente la calidad máxima obtenida. En términos de variabilidad, el caso más estable es de dimensión 5 y dificultad 1 pero a medida que aumenta la calidad se aumenta la variabilidad. Es decir, para obtener una calidad muy alta se requiere una combinación de parámetros que hace el algoritmo ser más inestable. Se puede comprobar que en el dimensión 5 y dificultad 5, algunos de los puntos obtienen una calidad muy baja, pero que están presenten en el gráfico debido a su baja variabilidad, por ser representaciones de los frentes de Pareto.

Se muestran en la parte inferior de la figura, las gráficas correspondientes a parámetros, de izquierda a derecha: c_m, T_{mat}, P_{ls}, S_{max}. Se puede observar que c_m tiene un comportamiento más claro que para la función anterior, ya que todos los puntos de calidad alta corresponden a $c_m = 0$, que quiere decir que los individuos con una calidad privada lejos de la óptima deben ser eliminados muy rápido de la población. También se ve una correspondencia entre los puntos de baja calidad en el caso de dificultad 5 y dimensión 5, que corresponden con valores altos de c_m. En cuanto al parámetro T_{mat}, al igual que para la función anterior, con T_{mat} igual a 1, independientemente del resto de parámetros, es cuando se obtienen mejores resultados. Hay que recordar que tanto T_{mat} como c_m afectan a la curva de tiempo de vida de los individuos y un c_m y T_{mat} bajo corresponden a eliminar individuos mediocres de la población lo antes posible.

El parámetro más interesante en este experimento es el parámetro P_{ls} debido al contraste con el resultado de este parámetro en la función anterior. En este caso se observa que la tendencia es la contraria a la anterior y ahora los puntos de mayor calidad corresponden a un P_{ls} alto. Además, con el incremento

de dificultad se hace más necesario incrementar el parámetro P_{ls} hasta 1 para obtener buenos resultados, lo cual quiere decir que para problemas complejos es necesario aplicar una búsqueda totalmente local. El parámetro S_{max} también se comporta diferente en esta función con respecto a la anterior. En este caso no existen casi puntos en los frentes de Pareto con S_{max} menor que 10, con lo que para esta función se requieren valores de S_{max} mayores que 10, independientemente del resto de parámetros, para resolver el problema. Cuando se reemplazan individuos mediocres de forma más frecuente, esto puede afectar negativamente a la selección para reproducción, que disminuye, pero el aumentar S_{max} compensa ese efecto.

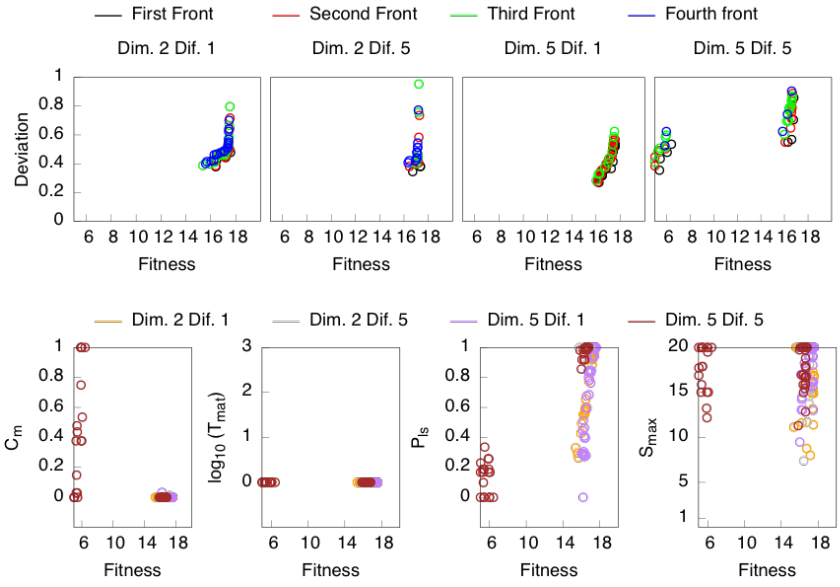


Figura 6.10: Resultados del método de innovization para la función 2. En la fila de arriba se muestran los frentes de Pareto para contener al menos 30 puntos. En la fila de abajo se muestra el comportamiento de los parámetros intrínsecos.

6.3.2.4. Función multi especie, pseudo separable (tarea óptima colaborativa)

Para la función global que contiene dos especies en la solución optima pero un subóptimo en los genotipos homogéneos (función 3), se muestran los resultados en la figura 6.3.2.4. En la parte superior se observa que para esta función el incremento de dificultad afecta mucho mas que para las anteriores funciones. Para una dificultad igual a 1, la calidad media se sitúa sobre 18, que es similar a la obtenida con la función anterior, con los individuos balanceados en las especies optimas. Pero al aumentar la dificultad a 5, hay un decremento en la calidad media obtenida y un incremento en la variabilidad. Esto puede ser debido a la complejidad del espacio de búsqueda, muchos individuos solo alcanzan la solución subóptima y la población no está balanceada como para la función anterior. El efecto de la dimensionalidad no es muy relevante, y las dos gráficas correspondientes a dimensionalidad 5 son muy similares a las de dimensionalidad 2.

En cuanto a los parámetros, para obtener una calidad alta se observan las mismas tendencias que para la función de dos especies. Es decir, las dos especies optimas solo se alcanzan cuando los parámetros son c_m igual a 0, T_{mat} igual 1 , P_{ls} igual a 1 y S_max mayor que 10. Para los valores de calidad media que no son máximos no se pueden extraer conclusiones, lo que hace esta función muy interesante para un análisis posterior exhaustivo de la respuesta y estabilidad del algoritmo.

6.3.2.5. Función multi especie, pseudo separable (tarea óptima no colaborativa)

La ultima función que ha sido analizada con el método de innovization contiene una especie en su solución optima pero una solución subóptima con dos especies (función 4). Los resultados se muestran en la figura 6.3.2.5. En este caso los frentes de Pareto muestran al igual que para el caso anterior, una gran diferencia entre la dificultad 1 y dificultad 5. En el caso más simple, la calidad media máxima alcanzada por el algoritmo es 18. Es interesante comparar esta gráfica a la de la función 1 de una especie, para ver que la existencia de dos especies subóptimas implican un decremento de la calidad máxima. Para la dificultad 5, para esta función el algoritmo no es capaz de obtener las soluciones

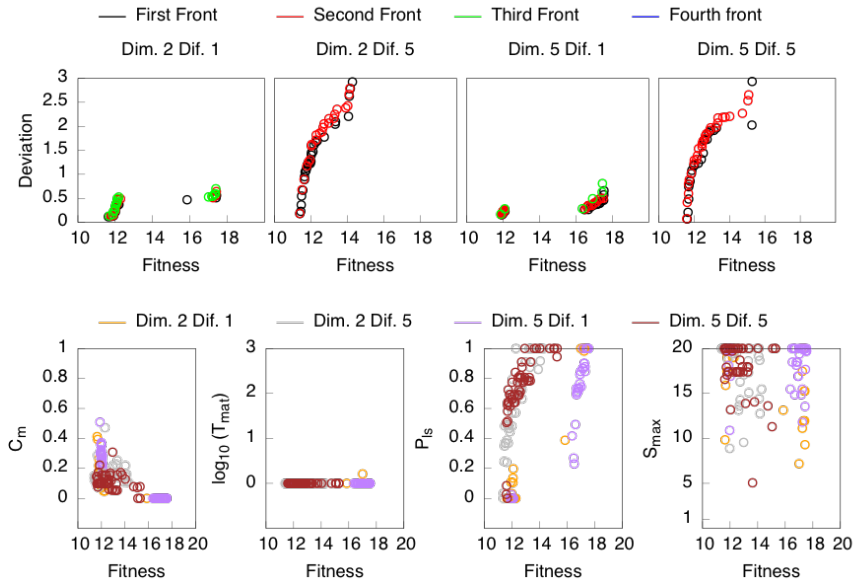


Figura 6.11: Resultados del método de innovization para la función 3. En la fila de arriba se muestran los frentes de Pareto para contener al menos 30 puntos. En la fila de abajo se muestra el comportamiento de los parámetros intrínsecos.

óptimas. La tendencia de los parámetros es muy interesante porque su comportamiento es mas similar al de la segunda función (dos especies óptimas) que a la primera función (un único óptimo), que no era esperado.

Igual que en las funciones anteriores, la solución óptima se alcanza solo usando un c_m en 0, un T_{mat} en 1, un P_{ls} cercano a 1 y un S_{max} mayor que 10. Con respecto el parámetro P_{ls}, ya que es una función con un óptimo homogéneo similar a la función 1, cabría esperar que los valores bajos de P_{ls} obtuvieran los valores de calidad media más altos pero se observa que en todo el rango de P_ls se obtienen valores de calidad alta, aunque con una ligera tendencia a valores más altos de calidad con P_{ls} más alto.

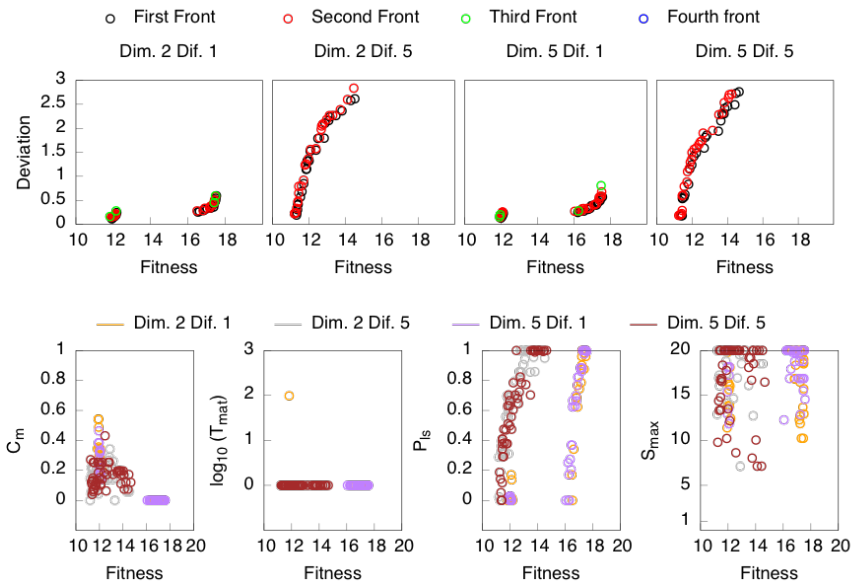


Figura 6.12: Resultados del método de innovization para la función 4. En la fila de arriba se muestran los frentes de Pareto para contener al menos 30 puntos. En la fila de abajo se muestra el comportamiento de los parámetros intrínsecos.

6.3.2.6. Conclusiones del preanálisis con el método de innovization

El método de innovization aplicado en esta sección ha proporcionado las siguientes conclusiones:

- Las funciones de calidad global más interesantes son las de múltiples especies y las pseudo separables (funciones 3 y 4), porque incluyen características de las otras dos y además corresponden a problemas colectivos más generales (con óptimos locales asociados tanto a poblaciones homogéneas como heterogéneas)
- Algunos de los parámetros afectan mucho al rendimiento del algoritmo, ya que algunas soluciones solo se obtienen cuando se encuentran en ciertos rangos o valores, con lo que pueden ser fijados en el análisis de sensibilidad exhaustivo del algoritmo canónico de dEE. Esto es un resultado muy relevante en cuanto a la caracterización del algoritmo. En particular, T_{mat} y P_{ls} se van a fijar a 1 para las siguientes secciones y sólo c_m y S_{max} se analizarán en profundidad.
- La sensibilidad del algoritmo canónico de dEE en términos de nivel de dificultad y dimensionalidad necesita ser analizada en más detalle.

6.3.3. Tests exhaustivos

Una vez que se han fijado dos de los parámetros (T_{mat} y P_{ls}), se puede analizar la respuesta del algoritmo realizando un barrido aleatorio en todo el rango para los parámetros c_m y S_{max}. La configuración del algoritmo para estas pruebas se muestra en la tabla 6.2. Como se ve, el rango del parámetro c_m se ha reducido de [0:1] a [0:0.33] para poder sacar conclusiones. Este es el rango en el que la mayoría de las configuraciones obtienen mayor calidad en el análisis con el método de innovization. Se han considerado tres dimensiones diferentes y para cada una de ellas, 4 niveles de dificultad (12 experimentos independientes) y para cada una de ellas.

En esta sección se mostrarán los resultados obtenidos con el algoritmo para las tres funciones de múltiples especies (función 2, 3 y 4). Para cada función de calidad, los parámetros c_m y S_{max} se muestrean aleatoriamente de forma uniforme en su rango, creando 2000 diferentes combinaciones de parámetros.

Tabla 6.2: Parámetros del algoritmo canónico de dEE para los tests exhaustivos

Configuración para test exhaustivos	Valor
Parámetro c_m	[0:0.33]
Parámetro T_{mat}	1
Parámetro P_{ls}	1
Parámetro S_{max}	[1:20]
Parámetro T_{max}	1000
Dimensionalidad	2, 5, 10
Dificultad	1, 3, 6, 10
Iteraciones por evolución	10000
Evoluciones por combinación de parámetros	25
Combinaciones de parámetros	2000

Cada una de estas combinaciones se usa para ejecutar el algoritmo canónico 25 veces con estos parámetros. La calidad global corresponde a la calidad media de la última mitad de las ejecuciones (5000 iteraciones), con el objetivo de evitar la etapa inicial del algoritmo.

Para simplificar el análisis de todas las pruebas, se usarán tres representaciones gráficas. En primer lugar, la respuesta del algoritmo se muestra a través de 12 gráficas de calidad frente a desviación, cada punto correspondiendo a una de las combinaciones. Esto permite extraer conclusiones sobre el rendimiento del algoritmo y la estabilidad con la parametrización propuesta. El mismo número de gráficas se usará para mostrar la calidad media en una escala de colores con los dos parámetros barridos enfrentados en cada eje, para visualizar las tendencias de los colores en los rangos de los parámetros.

Por último, para comprender mejor el comportamiento del algoritmo en términos de la configuración óptima de los genotipos en especies, se ha medido en las pruebas una calidad que se denomina calidad homogénea. Esta calidad representa la contribución a la calidad total de los individuos en un rango de genotipos no colaborativo. La diferencia entre la calidad total y la calidad homogénea representa la contribución a la calidad total de los individuos realizando la tarea colaborativa. La representación de la calidad homogénea es equivalente a la de la calidad total y puede ser comparada con las representaciones de calidad total para mostrar el comportamiento del algoritmo en términos de especialización.

6.3.3.1. Función de dos especies, no separable (tarea colaborativa)

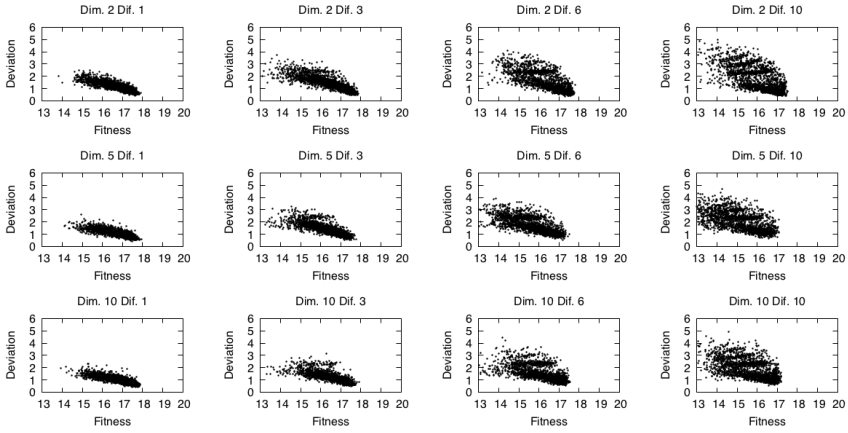


Figura 6.13: Calidad contra desviación para la función 2

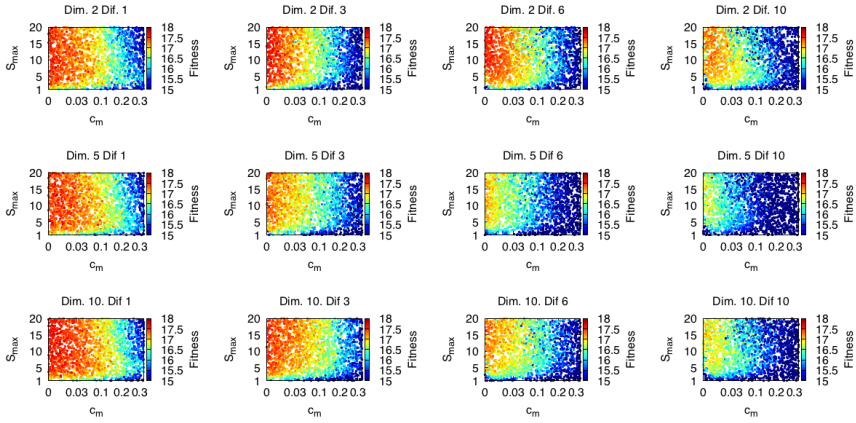


Figura 6.14: Calidad total para la función 2

Debido a que la primera función (función de una especie) no es muy compleja y el algoritmo obtiene soluciones óptimas en todo el rango de parámetros, las primeras pruebas exhaustivas que se muestran se han realizado para la función 2 (dos especies, no separable, tarea colaborativa). La figura 6.13 muestra las doce gráficas de calidad media frente a desviación. Cada uno de los puntos (2000

puntos en cada una de las doce gráficas) corresponde a una combinación de parámetros y se puede ver claramente el efecto de los parámetros que se están analizando, a pesar de estar fijados otros dos de los parámetros, ya que existen resultados en diferentes niveles de calidad desde 13 a 17. De izquierda a derecha se muestra como el aumento de dificultad afecta a la calidad media. En la figura 6.14 se muestra la calidad global para esta función. En cuanto al efecto de la dimensionalidad no se pueden extraer conclusiones generales ya que las gráficas para las diferentes dimensionalidades son muy similares.

En cuanto a los parámetros, en las doce gráficas el parámetro c_m se muestra en el eje horizontal y se puede ver una clara tendencia de valores de calidad más altos en la parte izquierda de las gráficas, que corresponden a c_m bajos. En cuanto al parámetro S_{max} se puede decir algo similar aunque el rango óptimo no es tan claro, aunque si se puede decir que para valores muy bajos de S_{max} menores que 5, se observan resultados con peor calidad. Como esta función es sólo de dos especies y no tiene ningún tipo de calidad para la especie homogénea, no se muestra la gráfica para esta calidad.

6.3.3.2. Función multi especie, pseudo separable (tarea óptima colaborativa)

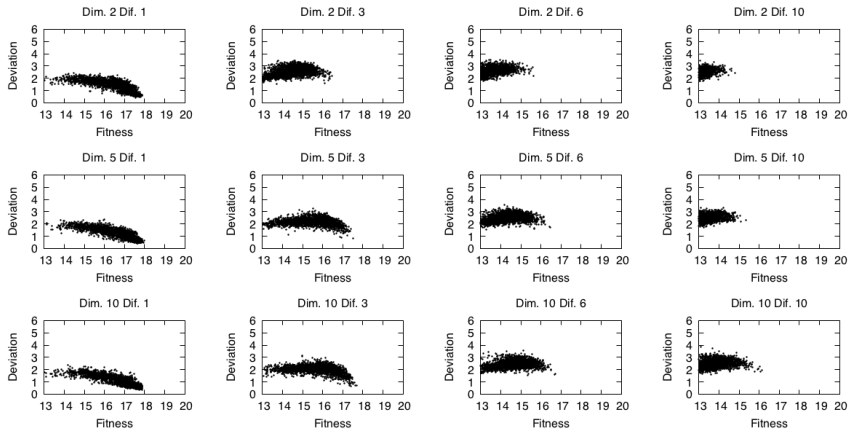


Figura 6.15: Calidad contra desviación para la función 3

En la figura 6.15 se muestran las doce gráficas de calidad media frente a

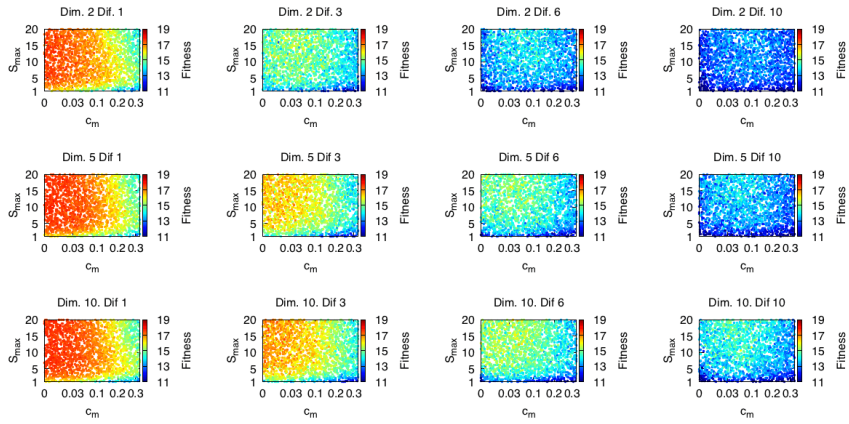


Figura 6.16: Calidad total para la función 3

desviación para la función 3 (multi especie, pseudo separable y que tiene un óptimo local en los genotipos homogéneos). Comparado con los resultados para la función anterior, para esta función el efecto de la dificultad es más notable.

En la figura 6.15, se muestra la gráfica que muestra la calidad y desviación resultante. Para dificultades altas (6 y 10), el máximo alcanzado por la calidad media es mucho menor que para la dificultad más baja. Al aumentar la dificultad también aumenta la desviación y los puntos se dispersan más en el eje horizontal. En cuanto a dimensionalidad se puede ver una tendencia de mejora de la calidad al aumentar la dimensionalidad en esta función. Para una dificultad fija por ejemplo, 10, si se observa la columna para esta dificultad, en la parte de la derecha, se observa claramente como los puntos están más desplazados hacia la derecha con calidad más alta a medida que aumenta la dimensionalidad. Intuitivamente se podría decir que el aumento de la dimensionalidad produce un aumento de la complejidad del espacio de búsqueda ya que suelen implicar una reducción del espacio óptimo, pero en estos experimentos la dimensionalidad no produce un óptimo más difícil de alcanzar sino que se mantiene constante el porcentaje de espacio de calidad que es óptimo. Es decir, al aumentar la dimensionalidad y aumentar el tamaño del espacio de búsqueda, se aumenta el tamaño de la zona del espacio óptima. De esta forma, el aumentar la dimensionalidad produce un mayor número de caminos que el algoritmo puede seguir para producir soluciones óptimas. De esta forma, se puede explicar este

incremento en calidad media al aumentar la dimensionalidad para esta función.

En cuanto a los parámetros para esta función, en la figura 6.16 se muestran las gráficas de la calidad total en función de los parámetros. Los resultados son similares a los producidos en la función anterior, excepto que la función al ser mas compleja obtiene valores menores de calidad incluso en cualquier zona de los parámetros para dificultad 10. En esta columna de dificultad prevalecen los tonos azules que son los de menor calidad. Para comprobar si los resultados para esta función están influenciados por el óptimo local, se muestra la misma gráfica pero para la calidad homogénea en la figura 6.17. En la primera columna para la dificultad 1, la calidad homogénea es muy baja, y tiene que ser así ya que esta función es óptima en las dos especies y en dificultad 1, la solución tiene una calidad muy alta. En la segunda columna sucede exactamente lo mismo, calidad homogénea muy baja en mientras que la calidad total correspondiente estaba muy alta. En la última columna, para dificultad 10, se observa algo relevante: la calidad homogénea está alta y corresponde en donde la calidad total no es alta, sino que es una calidad total mediana. En este caso, está demostrado que la calidad total corresponde a esta calidad homogénea y por lo tanto para esta dificultad, el algoritmo está atrapado en el óptimo local, lo cual no sucede para otras dificultades.

En cuanto a zonas relevantes de los parámetros, observando la gráfica de calidad total, se puede observar que prácticamente para cualquier configuración siempre hay que evitar valores de S_{max} bajos y valores de c_m altos, ya que siempre son los puntos de menor calidad en donde se concentran los tonos azules de baja calidad. Estas zonas se ven claramente para las dificultades más altas por ejemplo dificultad 6 y 10. Cuando S_{max} es mayor que 5, no se aprecia una diferencia significativa de valores de calidad ni ninguna tendencia. Que el algoritmo pueda obtener soluciones sin tener que fijar un S_{max} muy alto (5 se consideraría bajo), es una ventaja en términos prácticos, ya que en un experimento real puede ser difícil que un individuo encuentre muchos individuos para reproducción debido por ejemplo a ser un entorno muy grande.

6.3.3.3. Función multi especie, pseudo separable (tarea óptima no colaborativa)

La última función a analizar es la función 4, que es una función multi especie, pseudo separable y en donde el óptimo se alcanza con genotipos homogéneos

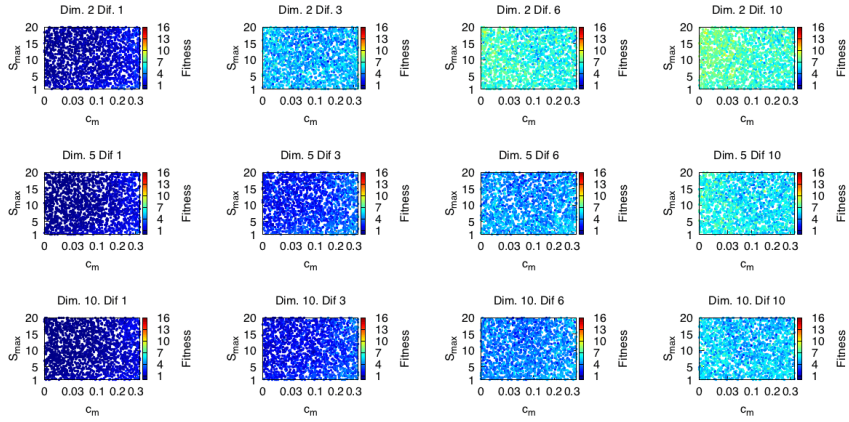


Figura 6.17: Calidad homogénea para la función 3

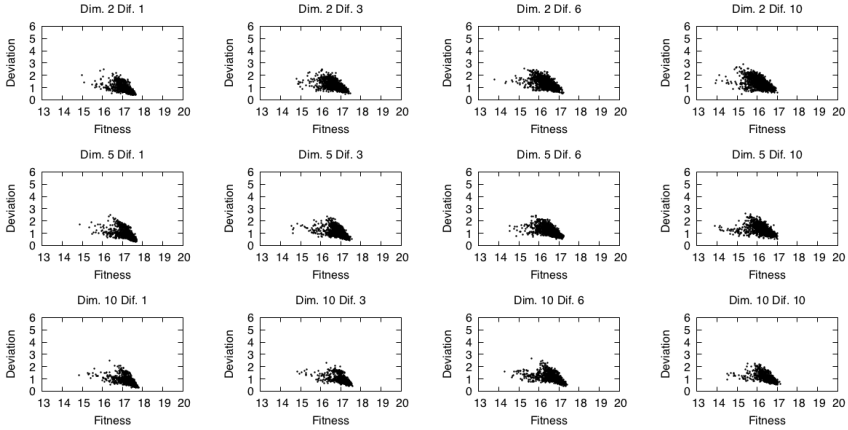


Figura 6.18: Calidad contra desviación para la función 4

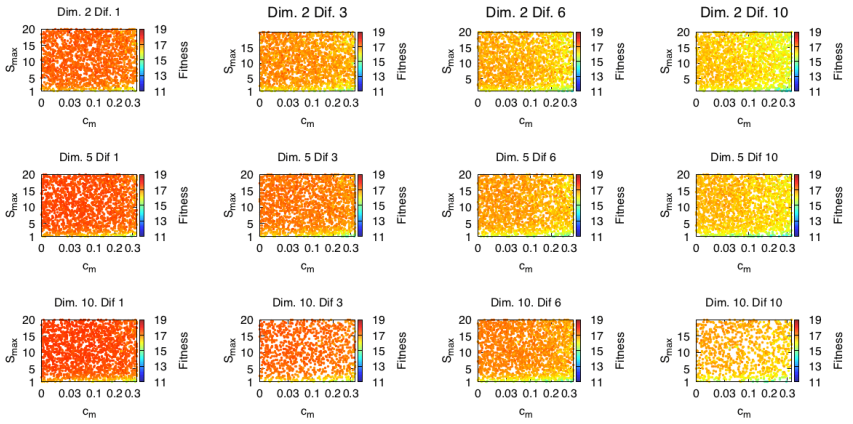


Figura 6.19: Calidad total para la función 4

pero tiene un máximo local con dos especies. Como para las funciones anteriores, en primer lugar se presenta en la figura 6.18 una gráfica representando la calidad media frente a la desviación. Lo primero a resaltar es que la función es más difícil que la función anterior, ya que se obtienen niveles superiores de calidad y con menor desviación.

En cuanto a los parámetros, en la figura 6.19, se muestran las gráficas resultantes para esta función. Para todo el rango de los parámetros los resultados son poco sensibles y lo único que se puede concluir es que la calidad es muy alta. Se comprueba también que para dificultad 10, se decrementa un poco el máximo de calidad media alcanzada. Si se compara esta gráfica con la gráfica de calidad homogénea que se muestra en la figura 6.20, se puede comprobar que esta calidad homogénea es también alta, lo que concuerda con el tipo de espacio de esta función, que es optimo con genotipos homogéneos.

6.3.4. Conclusiones de las pruebas teóricas

En este capítulo, se ha propuesto un conjunto de espacios de calidad basados en dos características: su separabilidad y número de óptimos. Estos espacios de calidad son representativos de problemas colectivos a los que se enfrenta un algoritmo de optimización distribuido y pueden ser utilizados como conjunto

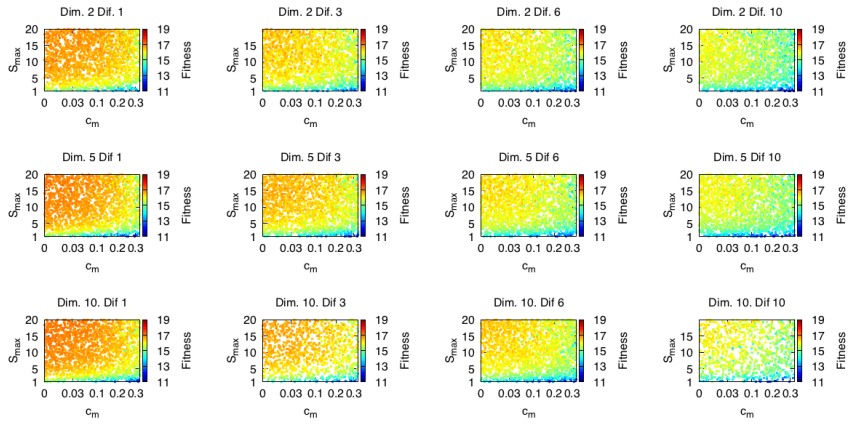


Figura 6.20: Calidad homogénea para la función 4

experimental de funciones de referencia para analizar este tipo de algoritmos. Sobre este conjunto experimental se ha comprobado la relevancia de los cinco parámetros intrínsecos propuestos para caracterizar al algoritmo canónico de dEE, primero con el método de innovization, y posteriormente con las pruebas exhaustivas.

El método de innovization ha servido para analizar cuales de las cuatro funciones eran las más relevantes para observar el efecto de los parámetros. Estas han sido la función 2 (no separable de dos especies óptimas), la función 3 (pseudo separable con dos especies en el óptimo y una especie homogénea en el óptimo local) y la función 4 (pseudo separable con dos especies en el óptimo local y una especie homogénea en el óptimo). Estas funciones fueron las utilizadas posteriormente en los análisis exhaustivos. Con el método de innovization, también se ha visto la sensibilidad de los parámetros especialmente de T_{mat} y P_{ls}, con zonas del rango de estos parámetros en los que el algoritmo no ofrecía buenos resultados. Específicamente, se ha comprobado que los individuos con poca calidad deben ser eliminados de la población lo antes posible y que el operador de recombinación genética que debe ser aplicado es la mutación, ya que es el que permite resolver todos los tipos de espacios de calidad y en particular las que necesitan especies colaborativas. El efecto de la diversidad de individuos para reproducción con S_{max} y el coeficiente de mediocridad c_m son relevantes pero sus efectos no son tan directos como los parámetros previos. Con los resultados

previos, se pudieron fijar T_{mat} y P_{ls} a unos valores con los cuales el algoritmo sí obtiene buenas soluciones ($T_{mat} = 1, P_{ls} = 1$), para de esta forma realizar unos análisis de sensibilidad exhaustivos del resto de parámetros (c_m y S_{max}).

En las pruebas exhaustivas se ha comprobado de nuevo la alta sensibilidad de los parámetros y se pudieron sacar unas conclusiones sobre los parámetros. Con respecto al parámetro c_m, las zonas en las que se obtienen mejores resultados dependen de la dificultad de la tarea pero un valor bajo de c_m es beneficioso para el algoritmo y con respecto al parámetro S_{max} debe ser lo más alto posible para que el algoritmo obtenga buenos resultados. Se deben evitar valores de c_m altos y valores de S_max bajos, ya que en estas condiciones el algoritmo obtiene peores resultados.

En resumen, el análisis exhaustivo nos da una pista a la hora de enfrentarnos a un nuevo problema de que parámetros hay que modificar en función de como sea el problema para que el algoritmo se beneficie de estos cambios y asimismo hemos visto también determinados valores que no son beneficiosos para el buen funcionamiento del algoritmo.

En el capítulo siguiente se validará el algoritmo canónico y su parametrización en un problema real en lugar de un conjunto de funciones sintéticas, tratando de verificar si las conclusiones extraídas de las pruebas teóricas siguen siendo válidas en un ejemplo práctico.

Capítulo 7

Caracterización del algoritmo cDEE en un problema real

Una vez que se ha caracterizado el algoritmo cDEE en un conjunto de funciones sintéticas, en este capítulo se va a analizar el algoritmo en un problema real con el objetivo de mostrar que la caracterización realizada es válida también en un problema real. Aunque es un problema particular se puede configurar para obtener los mismos casos de espacios de calidad que se analizaron en el capítulo anterior, por lo que se podrá comprobar si la parametrización del algoritmo realizada previamente es válida.

Específicamente, el problema planteado es una tarea de vigilancia colectiva con una degradación realista de la precisión de la localización de los robots debido a derivas en los sensores. El objetivo de esta tarea, como cualquiera de vigilancia, es explorar la mayor parte del entorno posible, pero la capacidad de exploración de los individuos depende directamente de su precisión en la localización. Es decir, el problema de la localización debe ser resuelto de forma implícita por el algoritmo para poder resolver satisfactoriamente la tarea. Esto configura un problema real de alto realismo práctico.

La navegación en escenarios interiores es un problema muy estudiado en

robótica autónoma. Para navegar de forma adecuada, el primer problema a resolver es localizar el robot en el escenario a partir de sus sensores de abordo, típicamente una IMU, una cámara, sensores de distancia, etc. Sin embargo, es conocido que la localización en espacios interiores es un problema complejo, principalmente porque no existen sistemas de referencia globales, la odometría sobre los sensores tiene una deriva y el uso de balizas naturales es aún un problema no resuelto muy estudiado en los últimos años [Davison et al., 2007] [Wendel et al., 2011] [Mur-Artal et al., 2015]. Cuando se desarrolla un sistema de navegación para un robot autónomo en un entorno real, la localización es clave en la respuesta del sistema de control del robot. Es evidente que la tarea implícita de localizar el robot es muy significante y condiciona el comportamiento óptimo de la tarea de navegación. Por lo tanto, es fundamental considerar la localización del robot como parte del problema si se pretende optimizar un sistema con aplicación real como el definido en el apartado 4.

En este experimento, la tarea consiste en un escenario simulado en el que una flota de MVAs (Micro Vehículos Aéreos) tienen que explorar colectivamente el entorno. Para realizar la tarea correctamente, los MVAs tienen que localizarse a sí mismos para conocer sus trayectorias y compartir esta información con otros robots para que cualquier robot tenga información de todo el entorno de una forma distribuida. La estimación de sus posiciones la realizan usando su IMU, marcadores artificiales que pueden ver con la cámara y con la posición de otros MVAs que observan, como se detallaran más adelante. El controlador de cada MVA se obtiene con el algoritmo cDEE, de forma que éste tiene que coordinar a los MVA en el entorno para maximizar el rendimiento de la flota. Como se puede comprobar, este experimento implica un objetivo principal, que es cubrir el mayor área posible, y está condicionado por un objetivo secundario, que es obtener suficiente precisión en la localización para navegar el entorno. Además, este problema es más general y más complejo que otros problemas típicos en robótica evolutiva, como puede ser buscar una fuente de luz o puntos de interés o recolectar trozos de comida dispersos en el entorno. En definitiva, el experimento cumple los requisitos para poder caracterizar al algoritmo cDEE en la práctica. Este capítulo se va a dividir en diferentes secciones: en primer lugar una configuración experimental que contiene como se ha formulado la tarea, a continuación se describirán varios experimentos realizados sobre esta tarea y por último se presentara una comparación del algoritmo cDEE contra otras aproximaciones sobre esta tarea.

7.1. Configuración experimental

El experimento se ha definido en simulación a partir de una tarea real de vigilancia en interiores realizada por MVAs. Concretamente, el MVA que se va a modelar es el Parrot ARDrone 2, un MVA de cuatro hélices comercial muy popular debido a su bajo coste. En la figura 7.1, a la izquierda se muestra una foto de este MVA visto desde arriba y a la derecha se muestra una foto en una prueba de vuelo con marcadores en un entorno real. Las sensores con los que cuenta este MVA son los siguientes:

- altímetro por ultrasonidos y medidor de presión
- unidad de medición inercial (IMU):
  - acelerómetro de tres ejes
  - giroscopio de tres ejes
  - magnetómetro de tres ejes
- cámara frontal con resolución 1280x720
- cámara hacia abajo con resolución 320x240

Los actuadores son cuatro motores "brushless" con hélices de 20 cm que permiten al MVA maniobrar en cualquier dirección. En el modelo real, para la estimación de la localización se utilizó un filtro de Kalman extendido (EKF) que integra las informaciones de los sensores y los comandos de control para estimar en cada paso de tiempo la localización del MVA. El vector de estado utilizado para el MVA está compuesto por las siguientes variables:

$$x(t) = (x, y, z, \dot{x}, \dot{y}, \dot{z}, \Phi, \Theta, \Psi, \dot{\Psi})^T \in \mathbb{R}^{10} \tag{7.1}$$

Dónde:

- x,y,z son las coordenadas del centro del MVA (en metros) en los distintos ejes del sistema de coordenadas correspondiente al mundo real
- $\dot{x}, \dot{y}, \dot{z}$ son las velocidades del MVA (en metros por segundo) expresadas en los distintos ejes del sistema

- Φ, Θ, Ψ son los ángulos (en grados) que representan la orientación del MVA respecto al sistema de coordenadas del mundo real
- $\dot{\Psi}$ es la velocidad de giro con respecto al eje z en grados por segundo

Y la función de transición de estado es:

$$\begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \Phi \\ \Theta \\ \Psi \\ \dot{\Psi} \end{pmatrix} + \delta t \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x}(x) \\ \ddot{y}(x) \\ \ddot{z}(x,u) \\ \dot{\Phi}(x,u) \\ \dot{\Theta}(x,u) \\ \dot{\Psi} \\ \ddot{\Psi}(x,u) \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \Phi \\ \Theta \\ \Psi \\ \dot{\Psi} \end{pmatrix} \tag{7.2}$$

El problema de utilizar únicamente la información de los sensores del MVA para la estimación de la posición es que tienen un ligero margen de error, pero que se van acumulando a largo del tiempo (deriva), lo que produce que las estimaciones de la localización no sean precisas. Para solucionar esto, se hace necesario una fuente de información de localización que no se degrade a lo largo del tiempo. Esta fuente de información han sido unos marcadores artificiales denominados AprilTag [Olson, 2011], que se pueden imprimir fácilmente y proveen una estimación de la posición de la cámara con respecto al marcador, aunque se degrada a medida que aumenta la distancia entre la cámara y el marcador. En la figura 7.1 se muestra una fotografía de uno de estos marcadores. Fusionando esta información de los marcadores artificiales con las de los sensores del MVA, el filtro EKF puede estimar la posición.

Para hacer un modelo de simulación de este experimento, la navegación en interior se realiza en estos MVA simulados como lo harían en el caso real, estimando su posición a partir de los datos de su IMU y su cámara. El aspecto más importante de la simulación es el modelo de respuestas de la estimación de la posición cuando el MVA se mueve. En primer lugar, la IMU proporciona señales de velocidad por cada grado de libertad, y esto tiene que ser integrado para producir el movimiento estimado. La estimación de la velocidad está modelada como una distribución normal centrada en la dato real de velocidad con su

correspondiente matriz de covarianza, como es habitual modelar en navegación real. Como, además de la IMU, la otra fuente de información son los marcadores artificiales, es necesario modelar en la simulación dichos marcadores AprilTag. La simulación realizada es un modelo realista en que la estimación de la posición proporcionada por los marcadores está basada en el modelo de precisión que se produjo en nuestro laboratorio usando un ARDrone 2.0 y marcadores AprilTag de 40cm de ancho y largo. Siendo $\vec{p}_{mva}$ el vector de posición del MVA, $\vec{p}_{marcador}$ el vector de posición del marcador, yaw el ángulo entre la cámara y el marcador, la función φ relaciona la varianza de la estimación $Var(\vec{p}_{mva})$ con la distancia relativa ($\| \vec{p}_{mva} - \vec{p}_{tag} \|$) y el ángulo (yaw) entre la cámara y el marcador:

$$Var(\vec{p}_{mva}) = \varphi(\| \vec{p}_{mva} - \vec{p}_{marcador} \|, yaw) \tag{7.3}$$

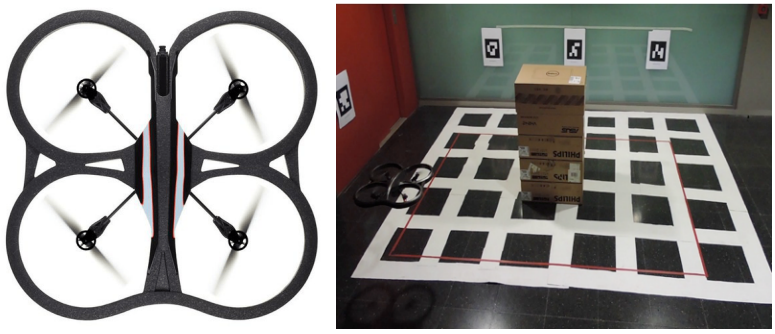


Figura 7.1: El Micro Vehículo Aéreo modelado: Parrot ARDrone 2.0. A la izquierda, vista desde arriba y a la derecha prueba de vuelo con marcadores en un entorno real

En el modelo real, los marcadores utilizados estaban fijos a una pared y proporcionaban una estimación absoluta y potencialmente precisa, que no se degrada con el tiempo pero que depende de la distancia entre el marcador y la cámara. En esta simulación, para mejorar el rendimiento de la navegación al mejorar la precisión de los MVAs, el mismo tipo de marcadores se sitúan sobre el cuerpo de los MVA, creando así marcadores móviles. La detección que proporcionan los marcadores móviles es una estimación directa de la posición, pero a diferencia de los marcadores fijos, la precisión de las estimaciones que proveen decrece a medida que decrece la precisión del MVA que lleva el marcador. La degradación de la precisión es proporcional a la velocidad del MVA, y como

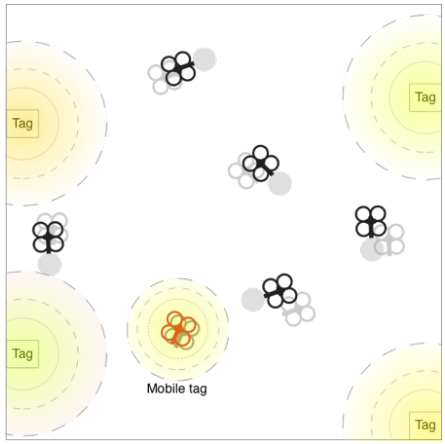


Figura 7.2: Representación gráfica de una parte del entorno para la tarea de vigilancia colectiva

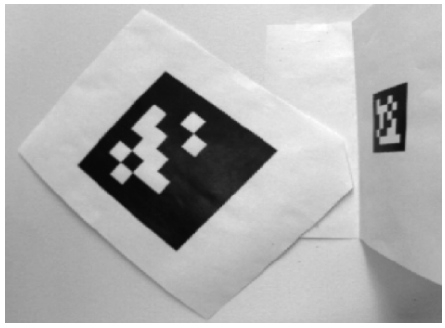


Figura 7.3: Fotografía de un AprilTag impreso

consecuencia, se ha definido en la tarea que los marcadores móviles deberían permanecer inmóviles para ser más eficientes como proveedores de precisión. El uso de este tipo de marcadores móviles supone que la precisión sea un recurso que los MVA puede obtener y compartir para frenar su degradación y en consecuencia, realizar su tarea más eficientemente. También podría permitir a algunos MVA no tener que visitar marcadores fijos que estén situados en lugares no óptimos, y poder incrementar su precisión solo provistos por marcadores móviles.

El escenario es cuadrado y no toroidal y cuenta con marcadores fijos situados aleatoriamente. En la figura 7.1 se muestra una representación gráfica de una parte del entorno. A efectos de visualización, se muestra cada MVA con su posición real y con su posición estimada. En esta figura, se muestra la posición real con un color sólido y la estimada como una sombra del MVA. La distancia entre esta sombra y la real representa el error de estimación. Cuanto mayor es este error de estimación, menor la capacidad de exploración. El MVA rojo representa un MVA actuando como marcador móvil. Los círculos alrededor de los marcadores representan diferentes niveles de precisión ofrecidos por el marcador. En la tabla 7.1 se muestran los parámetros específicos del entorno.

El objetivo final de la tarea de vigilancia es que la flota de MVA cubran el mayor área posible de una manera continua. Para poder realizar la búsqueda de las zonas sin explorar, los MVA almacenan las áreas que ya han explorado en un mapa de exploración que intercambian con otros MVA cuando los ven. El intercambio de esta información permite que la tarea sea cooperativa ya que permite la distribución de la búsqueda entre la flota. Sin embargo, debido a que las estimaciones de posición de cada uno de los MVA tienen diferente precisión, el correspondiente mapa de exploración de cada uno de ellos también. El entorno está dividido en celdas y la exploración de una celda se modela como una probabilidad de exploración (P_{ex}), que indica la probabilidad de que la celda haya sido explorada. Esta probabilidad se calcula como muestra la ecuación 7.4, como el ratio entre el área de la celda (L_{cell}) y el error de localización (E_{loc}), y es lo que se almacena en el mapa de exploración. Cuando otros MVA pasen por encima de una celda, tendrán que decidir si exploran de nuevo esa celda o no basándose en la probabilidad P_{ex}. Por lo tanto, el mapa de exploración es la única información que tienen los MVA sobre la tarea de vigilancia.

$$P_{ex} = \frac{L_{cell}^2}{\pi E_{loc}^2} \tag{7.4}$$

7.1.1. Definición de calidad privada y calidad global

La calidad global (Φ) definida para la tarea es la suma de las probabilidades de exploración P_e de cada una de las celdas del entorno, agrupadas en áreas, como muestra la ecuación 7.5. Estas áreas contienen el nivel medio de exploración de las celdas que se incluyen dentro de ellas, y sus dimensiones dependen de las divisiones del entorno. Dividiendo en áreas de tamaño 48x48, cada área contiene una única celda, en divisiones de tamaño 16x16, cada área tiene 9 celdas, en divisiones de tamaño 8x8 cada área tiene 36 celdas y en divisiones de tamaño 4x4 cada área tiene 144 celdas. Debido a que la probabilidad de exploración de cada una de las celdas decrece a una tasa constante, una probabilidad alta implica una exploración reciente en la celda. Los niveles de exploración varían desde 1, en el instante en el que una celda acaba de ser explorada por un MVA con la precisión máxima, a 0 si la celda no ha sido explorada en un periodo de tiempo. Por lo tanto, una calidad global alta en la tarea corresponde a una exploración rápida y exhaustiva de todo el entorno.

$$\Phi = \sum_{i=1}^{areas} P_{e,i,j} \tag{7.5}$$

La calidad privada (f_j) definida para los MVA se basa en la suma de incrementos de exploración que produce el MVA en cada celda que visita (Δe_i^j). Esta calidad no tiene en cuenta los individuos que colaboran con el desarrollo de la tarea pero no exploran el entorno, que son los individuos que proveen de precisión de localización (s_i^j, precisión compartida por el individuo i al individuo j). Para tener en cuenta este aporte de precisión, la calidad privada que los individuos obtienen al explorar es compartida con los individuos que les han proporcionado la precisión para realizar esa exploración, según una tasa (t_f). Por eso al incremento de exploración hay que restarle un porcentaje según esta tasa, que se le suma al individuo del que se obtuvo la precisión.

Teniendo en cuenta la definición anterior de calidad global que es la que va a maximizar el algoritmo, la función de calidad define un espacio de calidad

no separable, multimodal y con el optimo en dos especies, ya que se necesita la colaboración entre dos tareas diferentes (explorar y proveer precisión). No es posible realizar la tarea de explorar de forma óptima sin una fuente de precisión.

$$f_j = \sum_{i=0}^{visitadas} (\Delta e_i^j - t_f \Delta e_i^j) \tag{7.6}$$

7.1.2. Implementación del algoritmo cDEE

El pseudocódigo 6 muestra la implementación realizada del algoritmo cDEE para esta tarea real. Al igual que para las pruebas teóricas, se han utilizado los operadores de recombinación del algoritmo ASiCo.

Algorithm 6 Algoritmo cDEE

```
for all robot r en robots do
  actuar en el entorno
  calcular calidad privada
  P_mating ← S_max / T_max {calcular probabilidad de reproducción}
  P_rep ← σ(f_t) {calcular probabilidad de reemplazo}
  if random < P_mating then
    buscar individuos para reproducción
    seleccionar individuo {Parámetro}
    aplicar operadores genéticos de recombinación {Parámetro}
  end if
  if tiempo de vida ≥ T_mat then
    if random() < P_rep or tiempo de vida ≥ T_max then
      reemplazar individuo por nuevo
      controlador[r] ← nuevoindividuo
    end if
  end if
end for
```

7.2. Análisis del efecto de la degradación

Como se ha mencionado al comienzo de este capítulo, el objetivo de este ejemplo práctico es analizar la respuesta del algoritmo cDEE a diferentes espacios de calidad, al igual que se ha hecho para las funciones teóricas. En esta

tarea, la degradación de la precisión en la localización es un factor determinante para la resolución de la tarea, ya que la localización es un problema implícito que el algoritmo tiene que resolver para obtener resultados óptimos en la tarea. Para cumplir el objetivo propuesto, se realizará un estudio de diferentes configuraciones de esta degradación que proporcionan diferentes espacios de calidad ya que la tarea y su dificultad cambian completamente según este parámetro. Por ejemplo, en una configuración en la que los MVA no degradan su precisión de localización, no es necesario que realicen ninguna tarea de observación de marcadores. Es decir, se eliminan algunas de las subtareas en las que se podría descomponer el problema y se modifica por lo tanto el espacio de calidad como se ha realizado en las pruebas teóricas del capítulo 6. Las degradaciones de la precisión están definidas en base a la velocidad máxima de los MVA que es fija.

7.2.1. Definición del individuo

Los MVA están definidos por su posición espacial real $[x_r, y_r]$ (se asume que vuelan a una altura constante z_r) y su posición estimada (su orientación, dada por su ángulo de guiñada, se define por la dirección del movimiento y no se requiere explícitamente). También tienen una precisión estimada (a_e), que se describe como la desviación estándar de su posición estimada. La figura 7.4 ilustra el funcionamiento de los MVA. La cámara permite detectar los diferentes niveles de exploración del entorno, que tendría un color diferente según el nivel de exploración de cada celda. También permite detectar la distancia al marcador más cercano, en caso de estar en rango de observación de uno de ellos. La capacidad de exploración actual depende del error en la localización. La unidad de control es una red de neuronas artificiales de tipo "feed-forward" con tres neuronas en la capa de entrada, una neurona en la capa oculta y una neurona en la capa de salida. El genotipo a codificar para cada individuo es un vector de valores reales en el rango [-1,1] que corresponden con los parámetros de esta red. La salida de la red está discretizada en 5 zonas correspondientes a las 5 acciones predeterminas que establecen las salidas de los motores para realizar esas acciones. Estas acciones son: explorar en zona cercana, explorar en zona lejana, incrementar la precisión moviéndose hacia un marcador, compartir precisión y evitar marcador más cercano (movimiento en sentido opuesto).

Las diferentes configuración de la degradación que se estudiarán en este experimento para modificar el espacio de calidad son: degradación $V_{max}/16$,

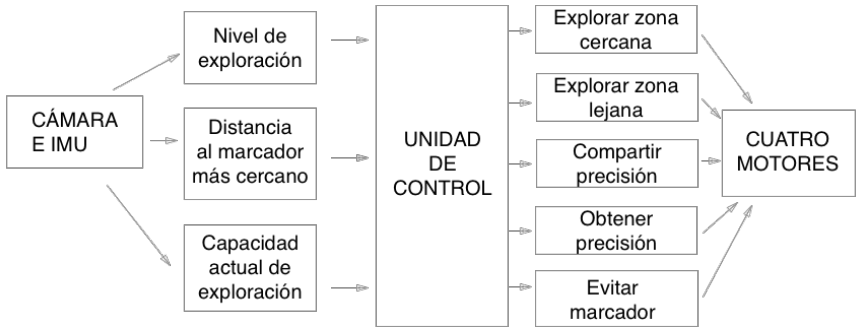


Figura 7.4: Esquema de funcionamiento de los MVA

Tabla 7.1: Parámetros del entorno simulado

Longitud del entorno (L)	768
Área total	L^2
Número de marcadores fijos	3
Rango de detección de marcadores fijos	L/4
Rango de detección de marcadores móviles	L/16
Ángulo máximo de detección de marcadores	$\pi/2$
Velocidad máxima (V_{max})	L/50
Máxima precisión proporcionada por un marcador	L/10
Error en precisión inicial	L/10
Degradación de precisión	$\{V_{max}/16, V_{max}/8, 0\}$

Tabla 7.2: Parámetros del algoritmo canónico de dEE para la tarea de vigilancia

Iteraciones	100000
Tiempo de vida máximo (T_{max})	1000
Tiempo de madurez	1
Criterio de selección	Mayor calidad
Tamaño máximo ventana de selección (S_{max})	40
Probabilidad de búsqueda local	0.99
Coeficiente de mediocridad	0.01
Calidad máxima actual	Automática
Tamaño de cromosoma	$8 \times [-1, 1]$

degradación $V_{max}/8$ y degradación 0, siendo V_{max} la velocidad máxima del MVA. Debido a que inicialmente los individuos parten con un error en precisión (L/10), siendo L la longitud de uno de los lados del entorno, en el caso de no tener degradación este nivel de precisión es suficiente como para realizar la tarea correctamente, y no necesitarían en ningún momento observar marcadores para mejorar su estimación de la posición.

Los parámetros del algoritmo canónico se muestran en la tabla 7.2 y concuerdan con las conclusiones sacadas del capítulo de pruebas teóricas para el espacio de calidad no separable con óptimo en dos especies heterogéneas: el tiempo de madurez se ha fijado a un valor muy bajo, el coeficiente de mediocridad también ha sido fijado a un valor muy bajo. La probabilidad de búsqueda local se ha fijado a un valor muy alto, que era el valor este parámetro que mejor resultados obtenía en esa función no separable, y el tamaño máximo de la ventana de selección se ha fijado a un valor alto. Dada esta configuración experimental se ejecutó el algoritmo cDEE para optimizar los controladores de 40 MVAs (el análisis de tamaño de equipo en las pruebas teóricas mostraban que 40 era adecuado) ejecutándose 25 veces para cada configuración de degradación y obteniendo una calidad global media de todas las ejecuciones. Los resultados se muestran en la figura 7.5, en donde se muestran en el eje horizontal las iteraciones y en el eje vertical el nivel de exploración media para las 25 ejecuciones. Este nivel de exploración media está medido en una división del entorno de 4x4 celdas, de forma que el máximo nivel de exploración teórico, si todas las celdas estuvieran totalmente exploradas, es 16. Es términos prácticos, debido a la extensión del entorno y el número de MVAs, un nivel de 13 sería óptimo. Como era de esperar, el problema sin degradación, que es el más sencillo, es el que obtiene mayor nivel de exploración, seguido por degradación Vmax/16 y Vmax/8. Es decir, a medida que la degradación aumenta y el problema se hace más complejo, disminuye el nivel de exploración alcanzado. Otro aspecto a tener en cuenta es el rápido incremento del nivel de exploración en el primer cuarto de todas las iteraciones, en donde el algoritmo es capaz de realizar un primer ajuste de la solución que luego se mantiene a lo largo del tiempo.

Se ha comprobado que los parámetros definidos para el algoritmo cDEE tienen una respuesta similar en la tarea, y el algoritmo obtiene buenas soluciones con los parámetros fijados según los criterios concluidos de las pruebas teóricas. Por ello, en este experimento es más interesante analizar las soluciones obtenidas en términos de especialización y comportamientos de los individuos según las

diferentes configuraciones del experimento.

Para analizar lo que ocurre en términos de los comportamientos que están seleccionando los individuos, se almacena la frecuencia en la que se selecciona cada uno de los comportamientos de cada individuo en cada paso de tiempo y con estas frecuencias se puede calcular la media de porcentaje de uso de cada uno de los comportamientos. Este porcentaje es lo que se representa en las siguientes figuras. En la figura 7.6, se muestra en una gráfica de áreas la proporción de cada uno de los comportamientos obtenidos para la configuración con degradación $V_{max}/16$. Como se puede ver el comportamiento predominante es la exploración en zona cercana, seguido del comportamiento de incrementar precisión. Otro comportamiento interesante es el de compartir precisión, que se vuelve relevante a partir de la mitad de las iteraciones. El resto de comportamientos: evitar marcador y explorar en zona lejana prácticamente no se realizan nunca. Este resultado es el esperado, ya que la función de calidad premia fundamentalmente explorar y compartir precisión actuando como marcador móvil. Las condiciones de degradación regularían la necesidad de más o menos proveedores de degradación pero este caso es una degradación media. La misma gráfica, para el caso con degradación $V_{max}/8$ se muestra en la figura 7.7. En este caso, el comportamiento de explorar en zona cercana no es tan predominante, y están más presentes que en el caso anterior los comportamientos de incrementar precisión y de compartir precisión. Esto es lo esperado, ya que debido al aumento de la degradación, el algoritmo adapta la solución con individuos que realizan con mayor frecuencia los comportamientos que aumentan la precisión. Los comportamientos que antes no eran muy frecuentes para este caso tampoco lo son, aunque aumenta la proporción para el comportamiento de explorar en zona lejana. Por último, la gráfica para la configuración sin degradación se muestra en la figura 7.8. Resulta clara la diferencia con las otras dos gráficas, ya que en esta configuración, al no necesitar aumento de precisión, la tarea con mayor frecuencia y prácticamente la única es la de exploración en zona cercana.

Por tanto, se ha mostrado el efecto de la degradación de la precisión en la localización para esta tarea, que modifica el espacio de calidad ya que se eliminan algunas subtareas necesarias para la tarea principal. La tarea principal es la de explorar pero para eso se requiere que algunos individuos de la población compartan precisión en una subtarea diferente. El algoritmo cDEE, al igual que se ha comprobado en funciones teóricas, es capaz de adaptar su solución a los diferentes espacios de calidad que presenta esta tarea de vigilancia colectiva. El

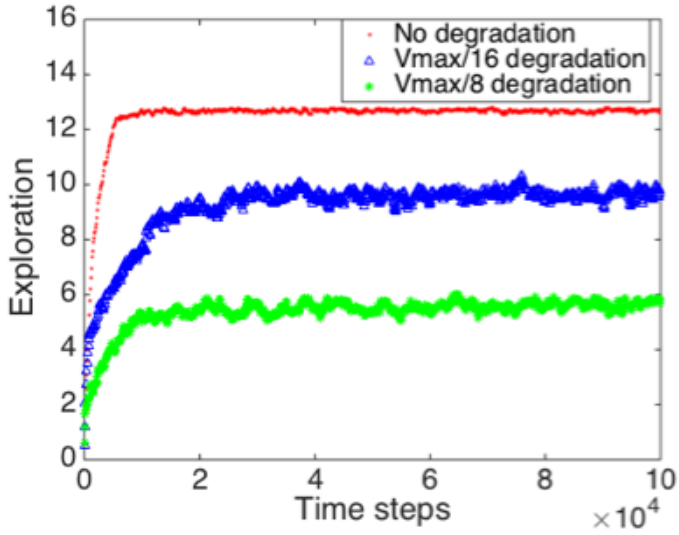


Figura 7.5: Calidad global media obtenida por el algoritmo con degradaciones Vmax/16, Vmax/8 y sin degradación

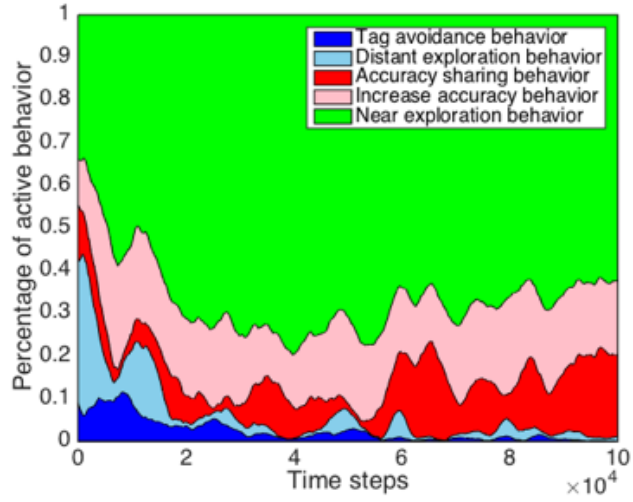


Figura 7.6: Proporción de porcentajes para degradación Vmax/16

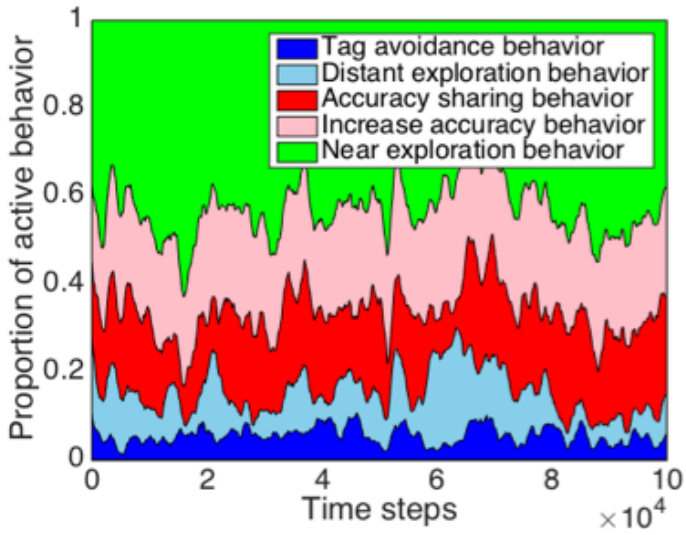


Figura 7.7: Proporción de porcentajes para degradación Vmax/8

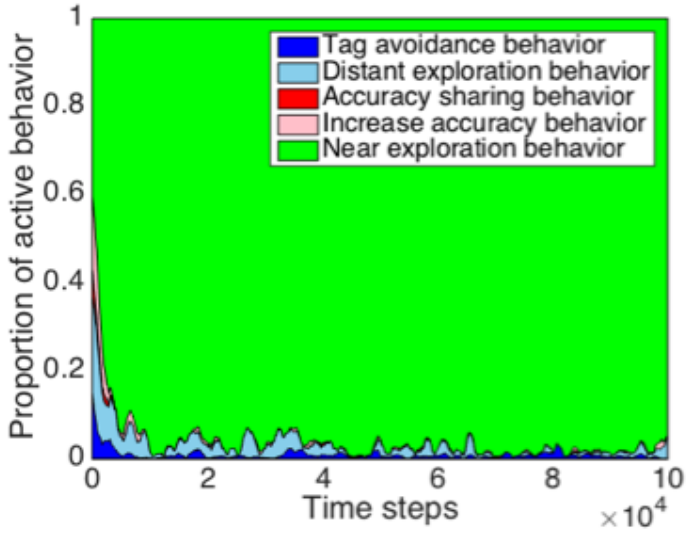


Figura 7.8: Proporción de porcentajes sin degradación

experimento confirma que la tarea sin degradación es una tarea como la función sintética de una especie ya que solo es necesaria un solo tipo de comportamiento que es explorar, mientras que la tarea con degradación se necesitan varios comportamientos, ya que los individuos no pueden explorar si no obtienen precisión para navegar.

7.3. Análisis de la complejidad del controlador

En este experimento, el controlador de los individuos se va a modificar para hacerlo más complejo y que permita mayor flexibilidad que el controlador del experimento anterior, que era una red de neuronas de tres capas con topología 3x1x1 y era fija. En este experimento se pretende estudiar el efecto de aumentar la complejidad de esta red en el rendimiento en la tarea. Debido a que una red más compleja, a priori, puede que realice más tareas a la vez que una red más simple, además de la complejidad del controlador, también se pretende analizar las soluciones en términos de especialización. Otro aspecto novedoso en este experimento era que en el caso anterior, no existían colisiones entre individuos ni contra los limites del entorno, lo cual simplificaba su comportamiento.

La complejidad del controlador se variará para este experimento aumentando el número de neuronas en la capa oculta, resultando la siguiente configuración de los individuos:

- sensores: además de detectar su capacidad de exploración actual (D_{ex}) y los niveles de exploración, ahora los MVA pueden detectar colisiones, tanto con otros MVA, como con los límites del entorno.
- Actuadores: están predefinidos para producir un movimiento en determinadas direcciones. Cuando los MVA seleccionan una celda objetivo, se mueven a velocidad máxima en la dirección correspondiente para aproximarse a esa celda.
- Controlador: la estructura de la red de neuronas ya no es fija y se va a experimentar con diferente número de neuronas en la capa oculta:
  - 1 peso en la capa oculta: red 3x1x1 de 4 pesos
  - 8 pesos en la capa oculta: red 3x2x1 de 8 pesos

Tabla 7.3: Parámetros del algoritmo canónico de dEE para el experimento de complejidad del controlador

Iteraciones	20000
Tiempo de vida máximo (T_{max})	1000
Tiempo de madurez	25
Criterio de selección	Mayor calidad
Tamaño máximo ventana de selección (S_{max})	40
Probabilidad de búsqueda local	1
Coeficiente de mediocridad	1
Calidad máxima actual	Automática
Tamaño de cromosoma	$\{4, 8, 40\} \times [0, 1]$

- 10 pesos en la capa oculta: red 3x10x1 de 40 pesos

Los cinco comportamientos predefinidos para este experimento son: explorar zona cercana, explorar zona lejana, incrementar precisión con un marcador, compartir precisión, evitar obstáculo. Se ha modificado el comportamiento del experimento anterior de evitar marcador cercano por evitar cualquier tipo de obstáculo que producirá que el MVA se mueva en dirección opuesta al obstáculo.

Para este experimento, se han modificado los parámetros del algoritmo, mostrados en la tabla 7.3 con respecto los del anterior experimento. Se ha aumentado el tiempo de madurez de 1 a 25 y se ha modificado el coeficiente de mediocridad hasta 1, debido a la complejidad de obtener los parámetros de la red (pesos) que son el espacio de búsqueda para el algoritmo. También se han modificado algunos de los parámetros del entorno que se muestran en la tabla 7.4, destacando que para este experimento la degradación es $V_{max}/8$, que era el caso de mayor dificultad en el experimento anterior. Además, en este experimento también se analizará la respuesta del algoritmo en un entorno en el que no existan marcadores fijos, sino que toda la información de precisión tenga que ser obtenida de individuos compartiendo su precisión. En esta configuración sin marcadores fijos, debido a que no existiría una fuente de precisión sin degradación, se considera que los marcadores móviles siempre tienen máxima precisión. Otra modificación en el entorno, con el fin de hacer más complicada la tarea, es que en el caso de colisionar con otro individuo o con las paredes del entorno, se penaliza con una pérdida de precisión en la localización a los individuos.

Tabla 7.4: Parámetros del entorno simulado para el experimento de complejidad del controlador

Longitud del entorno (L)	768
Área total	L^2
Número de marcadores fijos	{1, 0}
Rango de detección de marcadores fijos	L/4
Rango de detección de marcadores móviles	L/16
Ángulo máximo de detección de marcadores	$\pi/2$
Velocidad máxima (V_{max})	L/50
Máxima precisión proporcionada por un marcador	L/10
Error en precisión inicial	L/10
Degradación de precisión	$V_{max}/8$

Para analizar las soluciones en términos de especialización, se ha introducido una estimación del número de especies usando un indice denominado Davies-Bouldin (DBI) [Davies and Bouldin, 1979]. Como esta medida no tiene en cuenta la existencia de un único agrupamiento, aquí el mínimo de especies será 1, por introducir una especie auxiliar de 10 individuos. El número de especies (N_s) se obtiene sumando la inversa de su medida DBI asociada para cada uno de los posibles agrupamientos (de 1 a 10) como se muestra en la ecuación 7.7. Para tener en cuenta la especie auxiliar introducida, al cálculo anterior se le resta 1. El parámetro γ es un factor de escala fijado a 1 en este caso.

$$N_s = (\sum_{i=1}^{10} i * DBI(i)^{-\gamma}) - 1 \text{ con } \gamma{=}2 \tag{7.7}$$

Los resultados se muestran en la figura 7.9. La calidad global, al igual que para el anterior experimento, es la suma del nivel de exploración de todo el entorno. Cada una de las seis gráficas en la figura corresponde a la calidad media global y número de especies para esa de esa configuración en 25 ejecuciones. Hay que recordar que para esta degradación, se obtenía un nivel cercano a 5 en el experimento anterior, con lo cual, se puede interpretar que una calidad global cercana a 6 está cercana a la solución óptima. En esta gráfica, se presentan dos columnas, la columna izquierda corresponde a un entorno con marcadores fijos y la columna derecha a un entorno sin marcadores fijos. El eje x representa el número de especies y el eje y la calidad global en el escenario. Cada una de las filas representa una de las configuraciones: 4 pesos, 8 pesos y 40 pesos. Para cada

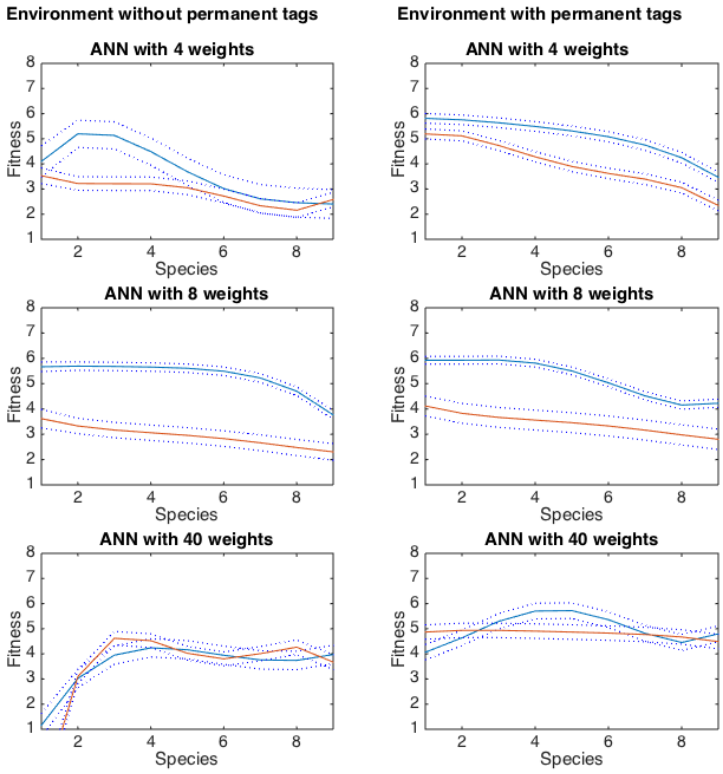


Figura 7.9: Efecto de la complejidad del controlador: 4 pesos, 8 pesos, 40 pesos

una de las configuraciones hay dos casos: un caso sin penalizaciones por colisión, que se ve en las lineas de color azul, y un caso con penalizaciones con colisión. La penalización por colisión consiste en aumentar el error de localización de los individuos y por lo tanto su capacidad de explorar.

Comenzando por los resultados obtenidos en esta gráfica por la configuración menos compleja de 4 pesos, en un entorno sin marcadores fijos, la calidad global máxima se obtiene con 2 o 3 especies, lo cual correspondería a individuos especializados en explorar y otros especializados en compartir precisión, ya que sino no se podría llevar a cabo la tarea, debido a la falta de marcadores fijos. El efecto de las penalizaciones en este caso es muy notable y no permite a los individuos realizar de forma óptima la tarea. Para esta configuración de 4 pesos, en el entorno con marcadores fijos, las calidades más altas se producen con un número reducido de especies, y es posible resolver la tarea con una especie obteniendo buena calidad, incluso en el caso del entorno con penalizaciones por colisión.

Si se observa la segunda fila con la configuración para 8 pesos, se puede ver que la calidad se mantiene en un valor alto para casi todos los valores de número de especies, excepto en valores de número de especies muy altos. De esto se podría interpretar que la red con esta complejidad es versátil y puede realizar la tarea tanto de forma especializada, como a través de una especie que ejecuta todos los comportamientos posibles. La calidad obtenida en el entorno con un marcador fijo, es similar a la calidad obtenida para la red de 4 pesos, excepto que el efecto de la penalización por colisión también es más notable que para la red de 4 pesos.

Para la configuración más compleja, la red de 40 pesos, los resultados en términos de calidad son mucho más bajos que para el resto de configuraciones y el algoritmo, en especial en el entorno sin marcadores fijos, tiene problemas para obtener la solución óptima. Esto puede ser debido a la dificultad del problema y a que es un espacio de alta dimensionalidad (40 individuos con 40 pesos cada uno). Sin embargo, con 40 pesos pero en un entorno con un marcador fijo, se obtiene una calidad alta tanto con penalizaciones por colisión o sin ellas.

Los resultados obtenidos en este análisis de la complejidad permiten concluir que una complejidad simple del controlador ya obtiene buenos resultados en la tarea y que no es necesario un controlador complejo. Los controladores complejos no ofrecen ninguna ventaja ya que el rendimiento obtenido con ellos es idéntico

a los controladores simples. Ademas, una desventaja de ellos, es que incrementa la dimensionalidad del problema para el espacio de búsqueda del algoritmo y por tanto también incrementa la dificultad de obtener el optimo.

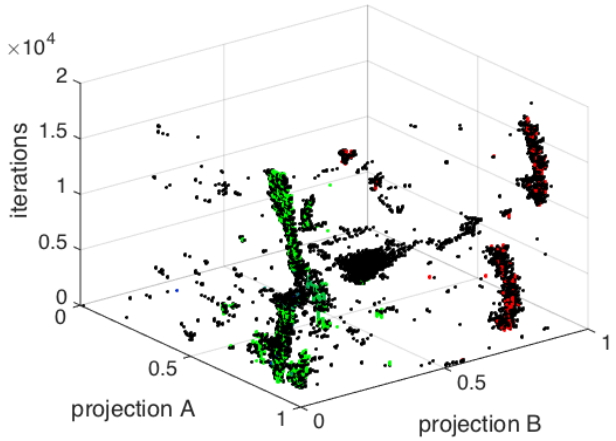


Figura 7.10: Representación de los genes de la población durante la evolución. Cada genotipo se proyecta en dos dimensiones independientes

Con el fin de comprobar la especialización que se produce en la solución obtenida por el algoritmo, se muestra en la figura 7.10 una representación de los genes de la población a lo largo de una evolución en una configuración de 40 individuos con genotipos de 4 pesos. Cada genotipo se proyecta en dos dimensiones independientes y en el eje z verticalmente, se muestran las iteraciones. El color de los puntos representa el comportamiento más frecuente del individuo con ese genotipo: verde para individuos que mayoritariamente realizan exploración y rojo para individuos que mayoritariamente realizan el comportamiento de proveer precisión como marcador móvil. Los individuos que tienen un tiempo de vida menor que 100 se muestran en color negro. Como se puede ver, desde una configuración inicial aleatoria la población converge rápidamente a dos especies, que son las agrupaciones de los genotipos. Los resultados obtenidos en esta representacion permiten comprobar que a pesar de representarse parámetros en el espacio de fenotipos de la red, existen como dos tipos de conjuntos de pesos en la población, por un lado, una especie que ejecuta el comportamiento de explorar y por otro lado, una especie que ejecuta el comportamiento de compartir

precisión. Esto concuerda con los resultados obtenidos en el capitulo de pruebas teóricas con la función no separable con optimo en dos especies heterogéneas.

Como conclusiones de este experimento, se puede mencionar que no se necesita una complejidad muy alta para resolver la tarea ya que el algoritmo es capaz de encontrar una solución especializada que ofrece una solución optima a la tarea con los controladores mas sencillos de 4 pesos. Aumentar la complejidad del controlador no ofrece ninguna ventaja en términos de calidad global pero implica un aumento de la dificultad de resolver el problema. Debido a esto, en el siguiente experimento se utilizará una red de 4 pesos.

7.4. Comparación con otras aproximaciones

Una vez que se ha validado la respuesta del algoritmo cDEE en esta tarea práctica, el último experimento que se plantea aquí es comparar el rendimiento del algoritmo cDEE con otras aproximaciones evolutivas similares. Como se mostró en el capítulo 3 la técnica de Embodied Evolution (EE) no es la única técnica evolutiva que se puede aplicar a problemas multi-robot y en esta sección se comparará el algoritmo canónico de dEE en esta tarea de vigilancia colectiva contra otras técnicas de evolución, en particular contra algoritmos de tipo CCEA (Cooperative Coevolution Evolutionary Algorithms) que tienen una estructura similar a los algoritmos de EE y que se han aplicado de forma satisfactoria a optimización de sistemas multi-robot, tal y como se analizó en el capítulo 3. De todos los posibles algoritmos con los que se podía comparar el algoritmo cDEE se han seleccionado varios para abarcar toda la gama de algoritmos evolutivos aplicados a sistemas multi-robot: un algoritmo CCEA que se ha denominado DECC (Differential Evolution Cooperative Coevolution) en dos variantes y un algoritmo encapsulado de Embodied Evolution con dos variantes.

7.4.1. Definición del individuo

- Sensores: los sensores son los mismos que para el experimento anterior (sensores de explorabilidad y precisión disponible).
- Actuadores: los actuadores son los mismos que para el experimento anterior

- Controlador: el controlador vuelve a ser una red de neuronas artificial con topología 3x1x1, que corresponde al controlador simple del apartado anterior.

7.4.2. Algoritmo cDEE

El primer algoritmo que se va a comparar en este experimento es el algoritmo cDEE. El pseudocódigo de este algoritmo para este experimento es el que se ha mostrado anteriormente en el pseudocódigo 6

7.4.3. Algoritmo DECC

Tras haber revisado la literatura de CCEAs para encontrar un algoritmo estado del arte en optimización de alta dimensionalidad, se ha seleccionado el algoritmo DECC (Differential Evolution Cooperative Coevolution), que es una adaptación del algoritmo DECC-G de Yang et al. [Yang et al., 2008a]. La motivación del algoritmo DECC-G es aplicar coevolución cooperativa para descomponer problemas no separables de alta dimensionalidad. La novedad de este algoritmo es que su estrategia de descomposición de variables en grupos tiene en cuenta la relación entre las variables y permite al algoritmo obtener buenos resultados en problemas no separables en los que otros algoritmos CCEA no los obtienen. La estrategia del algoritmo DECC-G consiste en realizar grupos de variables inderdependientes para mejorar la optimización usando también un algoritmo evolutivo estado del arte como el Differential Evolution (Differential Evolution (DE)) para optimizar cada componente. El uso de este algoritmo es una innovación ya que otros algoritmos de coevolución cooperativa previos utilizaban algoritmos genéticos simples. Otro algoritmo que extiende al algoritmo DECC-G es el algoritmo MLCC (Multilevel Cooperative Coevolutionary), [Yang et al., 2008b] que ajusta automáticamente el tamaño del grupo de la descomposición. El algoritmo DECC implementado aquí se basa en estos conceptos de estrategia de agrupamiento y recombinación con los operadores del DE para aplicarlos a la coordinación de un equipo de robots. El funcionamiento del algoritmo DECC, mostrado con sus dos variantes DECC-RG y DECC-FG en los pseudocódigos 7 y 8, es el siguiente:

1. En primer lugar, se parte de un vector con todas las variables del problema

que se dividirán en grupos, con cada grupo evolucionando en instancias aisladas de un algoritmo Differential Evolution. El número de componentes dependerá del tamaño de los grupos. En cada ciclo, el primer paso del algoritmo es permutar el vector de variables. Este paso puede ser omitido y los grupos serán asignados inicialmente y permanecerán fijos (DECC-FG, pseudocódigo 7) o puede ser asignados aleatoriamente en cada generación (DECC-RG, pseudocódigo 8)

2. Después de la permutación, se crean nuevas poblaciones del Differential Evolution con los correspondientes grupos de variables y se evolucionan durante un número de evaluaciones. Cada ciclo del algoritmo consiste en evolucionar cada uno de los componentes secuencialmente. Dado que cada componente tiene una población interna, se podrían realizar muchas combinaciones entre individuos pero aquí es un proceso secuencial con evolución off-line y evolucionando cada grupo de variables a su tiempo.

3. Para cada evaluación, las variables actuales se transfieren a los robots y éstos realizan la tarea durante un número determinado de iteraciones. Después de la evaluación, el entorno se reinicia. La evaluación de un grupo de variables produce una calidad global en la tarea.

4. Si los grupos de variables son fijos y asignados exactamente a los parámetros del controlador de un robot, la evaluación de un grupo de variables y la calidad privada del robot coincidirá. Esta versión del algoritmo se denomina aquí DECC-FG-Private.

5. Después de que un componente ha evolucionado un número de ciclos, las mejores variables para este grupo se modifican en el vector de variables, y el siguiente componente evoluciona con estas nuevas soluciones.

6. Por último, cuando todos los componentes han evolucionado, se ha completado un ciclo completo, y se se vuelven a crear nuevos grupos y se evolucionan todo los componentes otra vez hasta completar un número de ciclos determinado.

7.4.4. Algoritmo encapsulado de Embodied Evolution

El algoritmo encapsulado implementado aquí es similar a otros algoritmos encapsulados como el de Elfwing et al. [Elfwing et al., 2011] o el de Haasdijk et

Algorithm 7 DECC-FG

```
parametros[r * s] ← random() { r: robots, s: longitud genotipo}
mejoresParametros[s] ← 0
while ciclos ≤ ciclos máximos do
  inicializar r poblaciones de Differential Evolution
  for all población en poblaciones de Differential Evolution do
    while evaluaciones < evaluaciones máximas do
      generar nuevo individuo {vector de s parámetros}
      f_nuevo ← calidad global
      if f_nuevo > f_antecesor then
        reemplazar antecesor con nuevo
      end if
      if calidad global > mejor calidad global then
        mejor calidad global ← calidad global
        mejoresParametros ← nuevos parámetros
      end if
      parametros[posiciones] ← mejoresParametros
    end while
  end for
end while
```

al. [Haasdijk et al., 2010]. Se han seguido los principios de operación de estos algoritmos con cambios estructurales en la evaluación para hacer los resultados comparables con el algoritmo DECC. En otros algoritmos encapsulados el tiempo de evaluación se reparte entre los individuos de cada población, pero en esta implementación, un controlador con calidad alta tendrá más evaluaciones que un controlador de baja calidad. El pseudocódigo para este algoritmo se muestra en el pseudocódigo 9. Los pasos del funcionamiento del algoritmo son los siguientes

1. cada robot tiene una población de Differential Evolution que evoluciona independiente de las otras

2. antes de cada evaluación, se realiza un torneo entre los individuos de cada población para seleccionar el controlador a evaluar

3. algunas de las poblaciones se seleccionan para crear individuos nuevos

4. el equipo anteriormente seleccionado y estos nuevos individuos se evalúa durante un tiempo fijo

Algorithm 8 DECC-RG

```
parametros[r * s] ← random() { r: robots, s: longitud genotipo}
mejoresParametros[s] ← 0
while ciclos ≤ ciclos máximos do
  permutar vector de parámetros
  inicializar r poblaciones de Differential Evolution
  for all población en poblaciones de Differential Evolution do
    while evaluations < evaluaciones máximas do
      generar nuevo individuo {vector de s parámetros}
      f_nuevo ← calidad global
      if f_nuevo > f_antecesor then
        reemplazar antecesor con el nuevo
      end if
      if calidad global > mejor calidad global then
        mejor calidad global ← calidad global
        mejoresParametros ← nuevos parámetros
      end if
      parametros[posiciones] ← mejoresParametros
    end while
  end for
end while
```

5. si la evaluación global de los nuevos individuos es mejor que la de sus antecesores, se reemplazan por los nuevos

6. todos los individuos que formaron parte del equipo actualizan su calidad media con esta nueva evaluación

Hay que destacar un punto importante y es que no existe migración entre poblaciones, por lo tanto, el algoritmo funciona como un modelo de islas. La secuencia de operaciones en este algoritmo implica que este algoritmo es síncrono y off-line, ya que las evaluaciones son fijas en un periodo de tiempo.

Existen dos diferencias fundamentales entre el algoritmo encapsulado y los algoritmos DECC: por una parte, en el algoritmo encapsulado cada población del Differential Evolution corresponde siempre a un robot, con tantas poblaciones como robots, y un tamaño de grupo que corresponde al numero de parámetros del controlador de un robot, al contrario que en el algoritmo DECC-FG en donde el grupo es fijo pero su tamaño puede corresponder a varios robots. Por otra parte, en el algoritmo DECC, solo un grupo crea individuos nuevos al mis-

mo tiempo, pero en el encapsulado, más de una población puede crear nuevos individuos, que son las seleccionadas aleatoriamente. Esto quiere decir que este algoritmo debe converger mas rápidamente que el algoritmo anterior porque una porción más grande de las variables se modifican en cada evaluación. También hay que destacar que el número de poblaciones activas generando nuevos individuos no puede ser demasiado alto, porque se perjudica la estabilidad de las soluciones, ya que debido al torneo está garantizado que en las poblaciones que no están activas se están seleccionando individuos de buena calidad pero no es así con los individuos nuevos.

Una variación de este algoritmo encapsulado que se ha considerado también para la comparativa, es una versión asíncrona de éste. El funcionamiento de esta versión se muestra en el pseudocódigo 9. A diferencia del encapsulado síncrono, en la versión asíncrona, la evolución es on-line y los controladores se evalúan hasta que a los robots se les acaba la energía, en lugar de un número fijo de evaluaciones. Los nuevos individuos se generan cuando acaba esta evaluación y cuando se reemplaza un controlador no se afecta ni al entorno, que no se reinicia, ni a otros robots. Debido a que las poblaciones dentro del robot tienen muchos individuos y todos los individuos necesitan ser evaluados, en lugar de crear un nuevo individuo cada vez que un robot agota su energía, se realiza un torneo para seleccionar el siguiente controlador del robot.

7.4.5. Configuración del experimento

Se describe a continuación la configuración del experimento para la comparación de los algoritmos en la tarea de vigilancia colectiva. Los parámetros del entorno son los mismos que para los experimentos anteriores, mostrados en la tabla 7.1 y los parámetros de cada algoritmo se muestran en las tablas 7.5, 7.6, 7.7. El número total de iteraciones para todos los algoritmos es el mismo, 4×10^7, que es un tiempo suficiente para los off-line como el DECC, para obtener una solución óptima. La tarea se realiza con 40 robots que siguen la especificación del apartado anterior. En los algoritmos con evolución off-line (DECC y encapsulado) cada equipo se evalúa durante 1000 iteraciones y el entorno se reinicia a sus condiciones iniciales después de cada evaluación. Debido a que unos algoritmos reinician el entorno y otros no, la medida de calidad que se usará en cada una de las ejecuciones para cada algoritmo será la calidad más alta obtenida en el entorno. Posteriormente se calculará la calidad media (media de calidades

Algorithm 9 Algoritmo asíncrono encapsulado

```
inicializar tantas poblaciones de Differential Evolution como robots
calidad[robots][n] ← 0
indiceEnDE[robots] ← 0 {indices de individuo para formar equipos}
for all robot r en robots do
  controlador[r] ← poblacionesDE[robot][0]
end for
while ciclos ≤ ciclos máximos do
  for all robot r en robots do
    actuar
    f_t ← φ(f_t, f_{t-1}, f_{t-2}, ...) {evaluación de calidad}
    P_rep ← σ(f_t) {calcular probabilidad de reemplazo}
    if tiempo de vida ≥ T_mat then
      if tiempo de vida ≥ T_max or random() ≤ P_reemplazo then
        if f_t > fitness[r][index] then
          poblacionesDE[r][indexInDE[r]] ← controlador[r]
        end if
        calidad[r][indiceEnDE] ← θ(f_t, calidad[r][indiceEnDE[r]])
        siguienteControlador ← seleccionar en poblacionesDE[r]
        if random() < P_generate then
          siguienteControlador ← generarNuevoIndividuoDE
        end if
        controlador[r] ← siguienteControlador
      end if
    end if
  end for
end while
```

Algorithm 10 Algoritmo encapsulado

```
inicializar tantas poblaciones de Differential Evolution como robots
while ciclos ≤ ciclos máximos do
  elegir equipo con torneo
  generar nuevos individuos
  f_equipoNuevo ← evaluar equipo
  if f_equipoNuevo > f_anteriorEquipo then
    reemplazar antecesor con el nuevo
  end if
  actualizar indices antecesores
  actualizar calidad de los miembros del equipo
end while
```

Tabla 7.5: Parámetros del algoritmo DECC para el experimento de comparación de algoritmos

Parámetro	Valor
Poblaciones DE	40
Tamaño de grupo	4
DE: Población	10
DE: F	0.5
DE: CR	0.5
Evaluaciones en ciclo	100
Tiempo de evaluación	1000
Iteraciones en ciclo	4×10^6
Ciclos	10
Iteraciones totales	4×10^7

Tabla 7.6: Parámetros del algoritmo encapsulado de EE para el experimento de comparación de algoritmos

Parámetro	Valor
Poblaciones DE	40
DE: Población	10
DE: F	0.5
DE: CR	0.5
Tiempo de evaluación	1000
Iteraciones totales	4×10^7

máximas) para las 20 evoluciones de 4×10^7 iteraciones que se ejecutará cada algoritmo.

En la figura 7.11 se muestran los resultados para los seis algoritmos. La primera observación que se puede extraer es sobre las iteraciones necesarias para obtener soluciones estables, que es mucho menor en algoritmos de evolución online, como el asíncrono encapsulado y el algoritmo cDEE. Mejoran en términos de calidad muy rápidamente, y en aproximadamente 5 millones de iteraciones, que constituye el 12,5 % del total de iteraciones, obtienen una solución estable que se mantiene hasta el final. Por contra, los algoritmos de evolución offline (el encapsulado y las dos variantes de DECC) requieren al menos la mitad del total de iteraciones para obtener una solución estable. El algoritmo que ofrece mejor rendimiento de entre los offline es el algoritmo encapsulado, mientras que

Tabla 7.7: Parámetros del algoritmo cDEE para el experimento de comparación de algoritmos

Parámetro	Valor
Tiempo de vida máxima T_{max}	1000
Tiempo de madurez T_{mat}	1
Criterio de selección	F
Tamaño ventana de selección S_{max}	40
Probabilidad de búsqueda local P_{ls}	0.99
Tiempo de evaluación	1000
Coeficiente de mediocridad c_m	0.01
Iteraciones totales	4×10^7

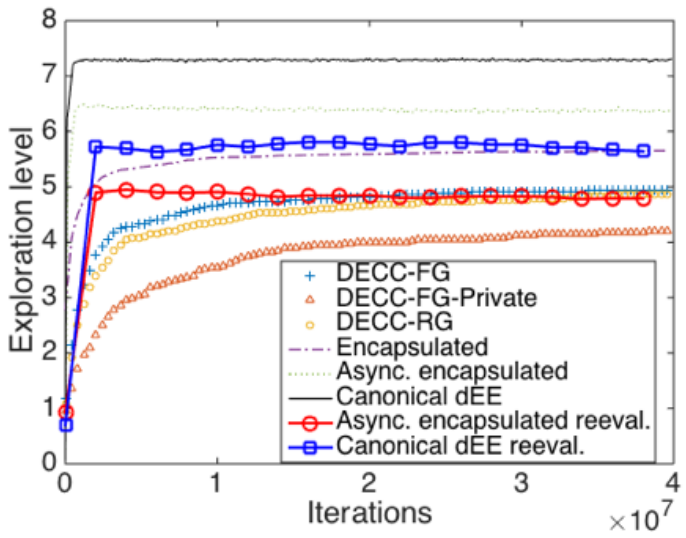


Figura 7.11: Comparativa de rendimiento de los diferentes algoritmos en la tarea de vigilancia

el DECC-FG ofrece mayor rendimiento que la otra variante DECC-RG.

Con el fin de comparar de forma justa los algoritmos online que no reinician el escenario, se guardan los equipos cada cierto número de iteraciones para evaluarlos posteriormente el mismo tiempo que las aproximaciones offline. Los niveles de calidad alcanzados en la reevaluación se muestran en la figura en la línea roja con círculos para el algoritmo cDEE y en la línea azul con cuadrados para el algoritmo asíncrono encapsulado. En la reevaluación obtienen un nivel de calidad estable similar al que ofrece el algoritmo encapsulado, pero aún mejor que las dos variantes de DECC. También se puede observar que, a pesar de llegar a un nivel similar al del algoritmo encapsulado, el algoritmo cDEE obtiene este nivel en un número menor de iteraciones. Además del nivel de calidad, otro análisis interesante que permite conocer el por que de cada uno de los resultados es comparar las soluciones aportadas por los algoritmos en términos de la heterogeneidad de los individuos que componen sus soluciones. La heterogeneidad se calcula individualmente a partir de los porcentajes de activación $(p_1, p_2, p_3, p_4, p_5)$ de cada una de las acciones siguiendo la formula siguiente. Una heterogeneidad 0 es que los individuos solo realizan una de las tareas, es decir, están especializados, mientras que una heterogeneidad 1 representa un individuo que alterna mucho su comportamiento.

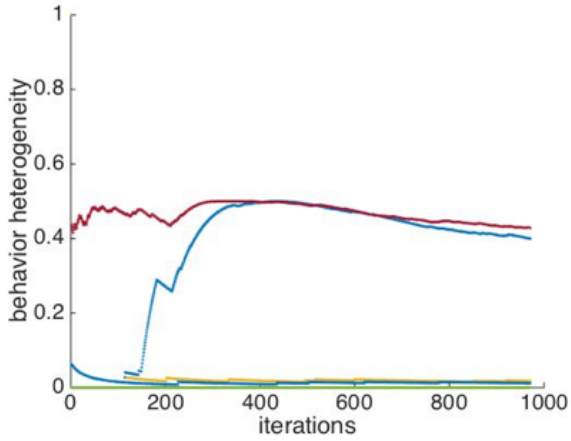


Figura 7.12: Heterogeneidad de comportamientos para el algoritmo cDEE

En la figura 7.12 se muestra la gráfica de heterogeneidad para los 40 indivi-

duos para el algoritmo cDEE. En el eje horizontal se muestran las iteraciones y en el eje vertical se muestra la heterogeneidad. Para este algoritmo, la mayoría de las heterogeneidades se sitúan superpuestas sobre el 0, ya que en total hay 40 líneas en la gráfica (correspondientes a 40 individuos). Se muestran dos líneas que son dos individuos que muestran un valor más alto de heterogeneidad. Teniendo en cuenta que este algoritmo obtuvo buena calidad global en la tarea, se comprueba que la causa de obtener esta calidad, es que los individuos están especializados en una tarea. En la figura 7.13 se muestra la heterogeneidad obtenida para el algoritmo encapsulado asíncrono. Este algoritmo obtenía peor calidad global que el algoritmo cDEE pero mejor que el resto de la comparativa, así que es el segundo mejor algoritmo en términos de calidad global. Se puede observar que también hay muchas heterogeneidades sobre el 0, pero ya existen más individuos con heterogeneidades altas que en el caso del algoritmo cDEE. Esto puede ser la causa de que la calidad no sea tan alta como en el caso anterior. Finalmente se muestra la gráfica de heterogeneidades para el algoritmo encapsulado en la figura 7.14, y el resultado es que existen heterogenidades en todo el rango, lo cual indica que no existe mucha especialización. Esto puede ser la causa de que este algoritmo obtuviera peor calidad que en los dos casos anteriores.

$$h_i = 1 - (p_1^2 + p_2^2 + p_3^2 + p_4^2 + p_5^2) \tag{7.8}$$

Una vez mostrados los diferentes experimentos en esta tarea práctica y los resultados obtenidos por el algoritmo cDEE, se puede concluir que se cumplido el objetivo de validar el algoritmo cDEE en un caso práctico ya que se obtienen soluciones óptimas. Estas soluciones corresponden a dos especies heterogéneas ya que la tarea se ha dividido en dos tareas y los individuos se especializan en realizar estas tareas. Esto corresponde a uno de los espacios de calidad planteados en el capítulo de pruebas teóricas.

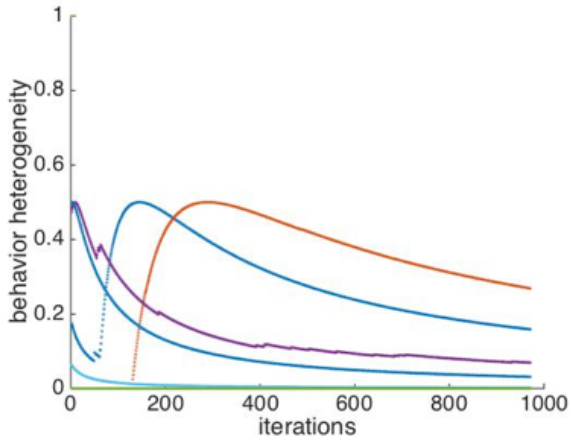


Figura 7.13: Heterogeneidad de comportamientos para el algoritmo encapsulado asíncrono

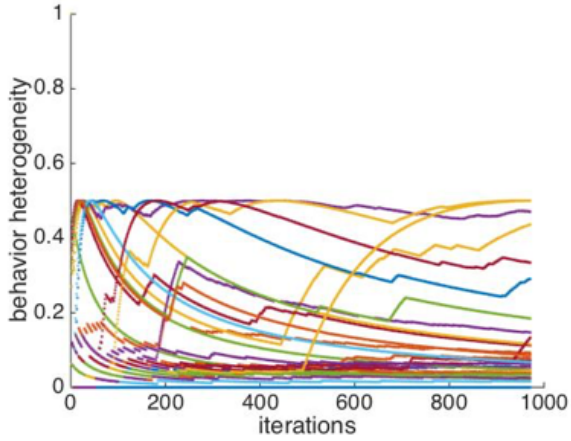


Figura 7.14: Heterogeneidad de comportamientos para el algoritmo encapsulado

Capítulo 8

Conclusiones

El objetivo principal de esta tesis, tal y como se estableció en el capítulo 2, ha sido el desarrollo de un algoritmo evolutivo para la optimización de sistemas multi-robot basado en el paradigma Embodied Evolution, que formalice las diferentes variantes que han surgido bajo esta aproximación. Dicho algoritmo se ha denominado algoritmo canónico de Embodied Evolution distribuido (cDEE) y con él se ha realizado un primer avance hacia la estandarización de este paradigma.

Este objetivo global parte de la motivación, surgida tras una dilatada experiencia en esta línea en el grupo de investigación del cual formo parte, de desarrollar una técnica de cómputo que permita el diseño automático de sistemas multi-robot que puedan ser transferidos a la realidad de manera simple. La revisión del estado del arte de las principales técnicas de optimización para sistemas multi-robot realizada en capítulo 3, muestra cómo ninguna de las aproximaciones existentes ha logrado un reconocimiento uniforme por parte de la comunidad científica, en unos casos debido a la excesiva complejidad que requiere el diseño manual de soluciones colectivas, y en otros porque la aplicabilidad real de las técnicas es limitada. Se debe destacar, eso sí, que las técnicas basadas en diseño automático mediante computación evolutiva han sido las que han proporcionado resultados más prometedores y por ese motivo se ha centrado dicho capítulo a su revisión detallada.

Por tanto, tras constatar la necesidad de dar un nuevo enfoque al problema

de la optimización de sistemas multi-robot, en el capítulo 4, se han establecido las especificaciones del tipo de sistema multi-robot final sobre el que se desea aplicar el algoritmo en el marco de esta tesis. Fundamentalmente, son tres: que debe estar formado por robots simples, que cada robot debe tener un alto grado de autonomía y no depender de un punto central, y por último, que sea posea un sistema de control adaptativo, capaz de soportar cambios en el entorno o en la tarea. De estas especificaciones para el sistema multi-robot han surgido una serie de requisitos de diseño para el algoritmo. Por un lado, debe permitir la optimización de sistemas multi-robot con independencia de su morfología, debe soportar evolución on-board de modo que no se dependa de ningún elemento central ni de un simulador, y finalmente, debe soportar evolución on-line para ser realmente adaptativo a cambios en tiempo real. Una técnica evolutiva que cumple todos estos requisitos es el paradigma de Embodied Evolution (EE), que surgió a finales de los años 90 para dar solución a la optimización de sistemas multi-robot en tiempo real, y sobre la cual se realizó un estudio de trabajos previos en la segunda parte del capítulo 4. Tras esta revisión del estado del arte, se pudo concluir, por un lado, la gran potencialidad de esta aproximación y el alto número de trabajos que se han venido desarrollando en las últimas dos décadas, y por otro, la falta de estandarización de los mismos. Así, han surgido numerosas variantes del algoritmo de EE original propuesto por Watson et al. pero con un grado muy bajo de formalización, centrando el esfuerzo en lograr resultados prácticos pero sin un análisis serio de su respuesta. En consecuencia, se plantea el desarrollo de una versión genérica del algoritmo EE basada únicamente en los procesos básicos de este paradigma y que además los parametrice de la forma más sencilla posible. En consecuencia, se establece la necesidad de desarrollar un algoritmo canónico de EE distribuido (cDEE), por ser esta opción más fiel que la encapsulada a la evolución natural en la cual se sustenta, tal y como se argumenta en la parte final del capítulo 4.

Para desarrollar el algoritmo cDEE, en el capítulo 5 se analizó el funcionamiento de los principales algoritmos en la aproximación distribuida, mEDEA, PGTA y el propio ASiCo, y se propuso un modelo teórico general para los tres procesos básicos: evaluación, reproducción y reemplazo. Estos procesos se han modelado mediante distribuciones de probabilidad genéricas y se han aislado de las particularidades del entorno y de la tarea, una de las principales aportaciones prácticas de esta tesis, ya que en todos los algoritmos analizados los procesos eran muy dependientes del dominio de aplicación. Para cada modelo probabilístico se estableció el número mínimo de parámetros que los controlan,

de tal forma que el cDEE sea fácilmente configurable. Finalmente, la última parte del capítulo 5 se dedicó a la implementación del pseudocódigo del cDEE.

A continuación, de acuerdo con los sub-objetivos planteados en el capítulo 2, se llevo a cabo un análisis del funcionamiento y de la sensibilidad paramétrica del cDEE. A la hora de realizarlo, se constató uno de los problemas de la robótica en general, también en el caso de sistemas multi-robot, que es la falta de un conjunto de tareas o pruebas de referencia que permita compara de manera formal las diferentes aproximaciones. Además de funciones de referencia, tampoco existe una estandarización en cuanto a la metodología a la hora de realizar dicha comparación. Una primer intento de poner solución a estos dos problemas se ha desarrollado en esta tesis, por una parte proponiendo un conjunto de funciones sintéticas que caracterizan a los espacios de búsqueda en problemas colectivos, y por otra, proponiendo una metodología para analizar la sensibilidad de los algoritmos en estas funciones. La definición de conjunto de funciones de prueba así como la metodología propuesta se han detallado en el capítulo 6.

En cuanto al conjunto de funciones sintéticas de referencia, las definidas en esta tesis representan la casuística de espacios de calidad más relevante que se presenta en problemas colectivos, en base a dos características: la separabilidad del problema (separable, no separable, pseudoseparable) y la combinación óptima de genotipos (una especie homogénea en el óptimo, dos especies heterogéneas en el óptimo, o presencia o no de óptimos locales). Estas funciones se parametrizaron con un factor de dificultad, que aumentaban o disminuían las zonas óptimas de los espacios de búsqueda. En la primera parte del capítulo 6 se llevó a cabo una introducción teórica sobre la relevancia de estas características en problemas colectivos, un estudio que, hasta nuestro conocimiento, no se había llevado a cabo en el campo con anterioridad.

En cuanto a la metodología propuesta, ésta tiene dos etapas: en primer lugar una etapa de pre-análisis con un método denominado “innovization”, que consistió en optimizar mediante un algoritmo multi-objetivo los parámetros del algoritmo cDEE ejecutado sobre las funciones de referencia, y en segundo lugar, una etapa de análisis exhaustivo con barridos en todo el rango de los parámetros sobre las mismas funciones. Para el pre-análisis con el método de “innovization” se visualizaron los frentes de Pareto para los parámetros, para comprobar qué valores de los parámetros producían los resultados con mayor calidad y menor variabilidad en las funciones de referencia. Este análisis se ha demostrado como muy útil, ya que se obtuvo mucha información acerca del comportamiento de

los parámetros, y a partir de esta información se pudieron fijar algunos de los parámetros para el análisis posterior. Además, este método podría ser aplicable a cualquier otro algoritmo en el caso de tener que tomar decisiones sobre el rango de unos parámetros en un algoritmo similar. El análisis exhaustivo posterior fue un barrido de todo el rango para dos de los parámetros y se visualizó la respuesta de estos dos parámetros en diferentes dificultades y dimensionalidades. El aumento o disminución de la dificultad se modeló como un estrechamiento o ensanchamiento de las zonas óptimas en los espacios de búsqueda mientras que la dimensionalidad se ajustó para que, a pesar de aumentar la dimensionalidad, las zonas óptimas del espacio de búsqueda fueran proporcionales en las diferentes dimensionalidades. Este análisis exhaustivo permitió determinar ciertos rangos de los parámetros que hay que evitar para obtener soluciones óptimas, como por ejemplo, con un tamaño de torneo de selección muy limitado, el algoritmo no puede obtener buenas soluciones.

Por último, tras analizar el algoritmo en funciones sintéticas, se necesitaba comprobar su validez en un caso práctico real, que es el contenido del capitulo 7. Concretamente se ha propuesto una tarea de vigilancia colectiva en la que era necesario resolver a su vez el problema de la localización autónoma de los robots. Esta tarea ha permitido, además de comprobar que el cDEE es capaz de optimizar un sistema multi-robot realista, estudiar diferentes propiedades del sistema como el tamaño de la población óptima, o diferentes condiciones del entorno que cambian el espacio de calidad. La configuración del experimento real se llevó a cabo mediante un parámetro de degradación de precisión en el escenario, que posibilitó la definición de diferentes espacios de calidad. La tarea de vigilancia sin pérdida de precisión se convirtió en un problema con una sola especie en el óptimo (solo se debe explorar), mientras que la tarea de vigilancia con pérdida de precisión representó un problema en el que dos especies heterogéneas debían colaborar para obtener un rendimiento óptimo. En esta tarea también se han experimentado con diferentes complejidades de los controladores de los robots, que ampliaban o reducían la dimensionalidad del espacio de búsqueda, y se mostró que el algoritmo obtiene soluciones óptimas para todas ellas. Finalmente, esta tarea ha servido para comparar el algoritmo cDEE con otros algoritmos similares, en concreto de tipo CCEA (Cooperative Coevolutionary Algorithm), referencia en la co-evolución de sistemas multi-robot pero que utilizan evolución off-line y off-board. De esta comparación se pudo extraer que el tiempo de convergencia para un algoritmo de Embodied Evolution es mucho menor, y que a pesar de obtener rendimientos similares con otros algoritmos,

las diferencias en coste computacional son tan amplias que harían que no tuvieran interés con respecto al paradigma de Embodied Evolution. La capacidad del algoritmo cDEE para obtener especialización en los individuos de manera automática según lo requiere la tarea, le proporciona una gran ventaja en tareas complejas al dividir el problema global en sub-problemas más simples.

Se debe concluir, finalmente, que el objetivo principal de esta tesis se ha logrado satisfactoriamente, y se ha desarrollado un algoritmo de gran potencialidad para la optimización de sistemas multi-robot que puedan ser transferidos con facilidad a la práctica. Se han dado pasos hacia la estandarización del paradigma Embodied Evolution, de tal forma que la comparación de diferentes aproximaciones sea más formal, y así se puedan establecer los límites de aplicación de esta técnica. Es evidente que se abre una línea de trabajo con numerosas vías de desarrollo, que el algoritmo cDEE debe ser mejorado y que el reto es ambicioso, pero el objetivo final de diseñar sistemas multi-robot que sean realmente aplicables seguirá siendo una gran motivación.

Capítulo 9

Trabajo futuro

El trabajo realizado en esta tesis ha permitido obtener un algoritmo canónico de embodied evolution (cDEE) que optimiza un sistema multi-robot en tiempo real y que se ejecuta de forma distribuida dentro de los robots. Durante el desarrollo y validación del algoritmo cDEE han aparecido nuevas oportunidades de mejora.

La principal línea de desarrollo futuro tiene que ver con la optimización de la morfología, uno de los aspectos básicos en la optimización de sistemas multi-robot y que tiene gran impacto sobre la optimización del control coordinado. Inicialmente, se deberá trabajar en la optimización del tamaño de la población, es decir, que el algoritmo funcione con un tamaño de población variable que debe ajustar. Hasta ahora, se ha fijado el número de robots que compone el sistema siguiendo criterios de diseño del algoritmo pero sin un estudio exhaustivo del tamaño ideal. Con un equipo muy numeroso, se favorecen las interacciones entre robots, y por tanto, la convergencia del algoritmo, pero también se incrementan las posibles interferencias entre ellos. lo que puede afectar negativamente a su rendimiento en la tarea.

Por otro lado, también se debe analizar la capacidad del algoritmo para optimizar la morfología de los robots, y no solo cuántos componen el equipo. Es cuando se consideran ambos aspectos cuando realmente se logra toda la flexibilidad del espacio de búsqueda, y donde el algoritmo evolutivo puede proporcionar resultados más adecuados. En este sentido, se podrá comenzar por evolucionar

el rango de los sensores y actuadores, por ejemplo, para tender hacia un diseño global del robot de manera similar a lo que se ha venido haciendo en el campo de la robótica modular.

En este ámbito, otra aspecto de mejora sería evolucionar la morfología de los controladores, por ejemplo, utilizando redes neuronales de tamaño variable. En esta tesis se ha realizado un estudio de la complejidad de los controladores que compara el rendimiento con varias complejidades, pero esta complejidad era fija y no variaba en tiempo de evolución. Codificando en los parámetros de robot la complejidad del controlador, de una forma similar a la que hacen otros algoritmos como el NEAT, se podría evolucionar de forma satisfactoria esta característica del controlador. Esta modificación no es muy directa, ya que hay que tener en cuenta que al cambiar la codificación de los individuos, también habría que cambiar los operadores genéticos para que funcionaran en esta nueva codificación.

En cuanto al proceso evolutivo que realiza el cDEE, en el fondo se sigue un proceso básico, con gran margen de mejora en los operadores. Así, por ejemplo, se podría tratar de adaptar un algoritmo Differential Evolution a Embodied Evolution, de tal forma que los operadores de cruce y mutación sigan el esquema de dicha aproximación. Este es un campo de desarrollo complementario con el anterior, ya que ambos van en la dirección de mejorar la capacidad de optimización del algoritmo que es la que define si se puede aplicar a casos complejos o no.

A pesar de que se ha caracterizado el algoritmo cDEE en unos parámetros intrínsecos, todavía es necesario fijar estos parámetros a priori antes de evolucionar. Sería deseable que el algoritmo ajuste automáticamente sus parámetros en tiempo real según corresponda, lo cual requiere un análisis previo del espacio de búsqueda al que se enfrenta.

Finalmente, un punto clave de trabajo futuro es la aplicación del algoritmo cDEE en tareas con coordinación más compleja y su ejecución en robots reales. Esta línea de trabajo es la que guía el resto de líneas comentadas, y las mejoras en el mismo dependen de su rendimiento práctico.

Referencias

[Agogino and Tumer, 2008] Agogino, A. and Tumer, K. (2008). Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2):257–288.

[Alami et al., 1998] Alami, R., Fleury, S., Herrb, M., Ingrand, F., and Robert, F. (1998). Multi-robot cooperation in the martha project. *IEEE Robotics & Automation Magazine*, 5(1):36–47.

[Ampatzis et al., 2009] Ampatzis, C., Tuci, E., Trianni, V., Christensen, A. L., and Dorigo, M. (2009). Evolving self-assembly in autonomous homogeneous robots: Experiments with two physical robots. *Artificial Life*, 15(4):465–484.

[Asada et al., 1999] Asada, M., Uchibe, E., and Hosoda, K. (1999). Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. *Artificial Intelligence*, 110(2):275–292.

[Baldassarre et al., 2003] Baldassarre, G., Nolfi, S., and Parisi, D. (2003). Evolving mobile robots able to display collective behaviors. *Artificial life*, 9(3):255–267.

[Bianco and Nolfi, 2004] Bianco, R. and Nolfi, S. (2004). Toward open-ended evolutionary robotics: evolving elementary robotic units able to self-assemble and self-reproduce. *Connection Science*, 16(4):227–248.

[Bongard, 2000] Bongard, J. C. (2000). The legion system: A novel approach to evolving heterogeneity for collective problem solving. In *European Conference on Genetic Programming*, pages 16–28. Springer.

[Botelho and Alami, 1999] Botelho, S. C. and Alami, R. (1999). M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1234–1239. IEEE.

[Brambilla et al., 2013] Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.

[Bredeche and Montanier, 2010] Bredeche, N. and Montanier, J.-M. (2010). Environment-driven embodied evolution in a population of autonomous agents. In *International Conference on Parallel Problem Solving from Nature*, pages 290–299. Springer.

[Bredeche et al., 2012] Bredeche, N., Montanier, J.-M., Liu, W., and Winfield, A. F. (2012). Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):101–129.

[Brooks, 1986] Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23.

[Cao et al., 1997] Cao, Y. U., Fukunaga, A. S., and Kahng, A. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, 4(1):7–27.

[Chand and Carnegie, 2013] Chand, P. and Carnegie, D. A. (2013). Mapping and exploration in a hierarchical heterogeneous multi-robot system using limited capability robots. *Robotics and autonomous Systems*, 61(6):565–579.

[Claus and Boutilier, 1998] Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, (s 746):752.

[Crespi et al., 2008] Crespi, V., Galstyan, A., and Lerman, K. (2008). Top-down vs bottom-up methodologies in multi-agent system design. *Autonomous Robots*, 24(3):303–313.

[Davies and Bouldin, 1979] Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227.

[Davison et al., 2007] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6).

[De Jong, 2006] De Jong, K. A. (2006). *Evolutionary computation: a unified approach*. MIT press.

[Deb and Srinivasan, 2006] Deb, K. and Srinivasan, A. (2006). Innovization: Innovating design principles through optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1629–1636. ACM.

[Doncieux et al., 2015] Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. G. (2015). Evolutionary robotics: What, why, and where to. *Frontiers in Robotics and AI*, 2:4.

[Dorigo et al., 2013] Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., et al. (2013). Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71.

[Dorigo et al., 2004] Dorigo, M., Trianni, V., Şahin, E., Groß, R., Labella, T. H., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D., et al. (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17(2-3):223–245.

[Duarte et al., 2016] Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., and Christensen, A. L. (2016). Evolution of collective behaviors for a real swarm of aquatic surface robots. *PloS one*, 11(3):e0151834.

[Dudek et al., 2002] Dudek, G., Jenkin, M., and Milios, E. (2002). A taxonomy of multirobot systems. *Robot teams: From diversity to polymorphism*, pages 3–22.

[Duro et al., 2010] Duro, R. J., Graña, M., and de Lope, J. (2010). On the potential contributions of hybrid intelligent approaches to multicomponent robotic system development. *Information Sciences*, 180(14):2635–2648.

[Eiben et al., 2010] Eiben, A., Haasdijk, E., and Bredeche, N. (2010). Embodied, on-line, on-board evolution for autonomous robotics. *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution.*, 7:361–382.

[Eiben et al., 2006] Eiben, A. E., Nitschke, G. S., and Schut, M. C. (2006). Collective specialization for evolutionary design of a multi-robot system. In *International Workshop on Swarm Robotics*, pages 189–205. Springer.

[Elfwing et al., 2005] Elfwing, S., Uchibe, E., Doya, K., and Christensen, H. I. (2005). Biologically inspired embodied evolution of survival. In *2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2210–2216. IEEE.

[Elfwing et al., 2011] Elfwing, S., Uchibe, E., Doya, K., and Christensen, H. I. (2011). Darwinian embodied evolution of the learning ability for survival. *Adaptive Behavior*, 19(2):101–120.

[Faíña et al., 2013] Faíña, A., Bellas, F., López-Peña, F., and Duro, R. J. (2013). Edhmor: Evolutionary designer of heterogeneous modular robots. *Engineering Applications of Artificial Intelligence*, 26(10):2408–2423.

[Farinelli et al., 2004] Farinelli, A., Iocchi, L., and Nardi, D. (2004). Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2015–2028.

[Ferrauto et al., 2013] Ferrauto, T., Parisi, D., Di Stefano, G., and Baldassarre, G. (2013). Different genetic algorithms and the evolution of specialization: A study with groups of simulated neural robots. *Artificial life*, 19(2):221–253.

[Floreano and Nolfi, 1997] Floreano, D. and Nolfi, S. (1997). God save the red queen! competition in co-evolutionary robotics. In *2nd Conference on Genetic Programming*, number LIS-CONF-1997-002.

[Gerkey and Mataric, 2002] Gerkey, B. P. and Mataric, M. J. (2002). Sold!: Auction methods for multirobot coordination. *IEEE transactions on robotics and automation*, 18(5):758–768.

[Gerkey and Matarić, 2004] Gerkey, B. P. and Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954.

[Gomes et al., 2013] Gomes, J., Urbano, P., and Christensen, A. L. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intelligence*, 7(2-3):115–144.

[Gomez and Miikkulainen, 1997] Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(3-4):317–342.

[Grabowski et al., 2000] Grabowski, R., Navarro-Serment, L. E., Paredis, C. J., and Khosla, P. K. (2000). Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308.

[Haasdijk et al., 2014] Haasdijk, E., Bredeche, N., and Eiben, A. (2014). Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics. *PloS one*, 9(6):e98466.

[Haasdijk et al., 2010] Haasdijk, E., Eiben, A., and Karafotias, G. (2010). On-line evolution of robot controllers by an encapsulated evolution strategy. In *IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE.

[Hamann et al., 2010] Hamann, H., Stradner, J., Schmickl, T., and Crailsheim, K. (2010). A hormone-based controller for evolutionary multi-modular robotics: From single modules to gait learning. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE.

[Haynes et al., 1995] Haynes, T., Sen, S., Schoenefeld, D., and Wainwright, R. (1995). Evolving a team. In *Working notes for the AAAI symposium on genetic programming*, pages 23–30.

[Heinerman et al., 2016] Heinerman, J., Zonta, A., Haasdijk, E., and Eiben, A. (2016). On-line evolution of foraging behaviour in a population of real robots. In *European Conference on the Applications of Evolutionary Computation*, pages 198–212. Springer International Publishing.

[Holland and Melhuish, 1999] Holland, O. and Melhuish, C. (1999). Stigmergy, self-organization, and sorting in collective robotics. *Artificial life*, 5(2):173–202.

[Kalra et al., 2005] Kalra, N., Ferguson, D., and Stentz, A. (2005). Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 1170–1177. IEEE.

[Karafotias et al., 2011] Karafotias, G., Haasdijk, E., and Eiben, A. E. (2011). An algorithm for distributed on-line, on-board evolutionary robotics. In *Pro-*

ceedings of the 13th annual conference on Genetic and evolutionary computation, pages 171–178. ACM.

[Kitano et al., 1997] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E. (1997). Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM.

[Kok et al., 2005a] Kok, J. R., Hoen, E. J., Bakker, B., and Vlassis, N. (2005a). Utile coordination: Learning interdependencies among cooperative agents. In *EEE Symp. on Computational Intelligence and Games, Colchester, Essex*, pages 29–36.

[Kok et al., 2005b] Kok, J. R., Spaan, M. T., and Vlassis, N. (2005b). Non-communicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*, 50(2):99–114.

[König et al., 2008] König, L., Jebens, K., Kernbach, S., and Levi, P. (2008). Stability of on-line and on-board evolving of adaptive collective behavior. In *European Robotics Symposium 2008*, pages 293–302. Springer.

[Kube and Zhang, 1993] Kube, C. R. and Zhang, H. (1993). Collective robotics: From social insects to robots. *Adaptive behavior*, 2(2):189–218.

[Lee et al., 2008] Lee, D., Seo, S., and Sim, K. (2008). Online evolution for cooperative behavior in group robot systems. *International Journal of Control Automation and Systems*, 6(2):282.

[Lehman and Stanley, 2008] Lehman, J. and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336.

[Lehman and Stanley, 2010] Lehman, J. and Stanley, K. O. (2010). Revising the evolutionary computation abstraction: minimal criteria novelty search. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 103–110. ACM.

[Levi and Kernbach, 2010] Levi, P. and Kernbach, S. (2010). Symbiotic multi-robot organisms: Reliability, adaptability. *Evolution*, 7.

[Littman, 2001] Littman, M. L. (2001). Value-function reinforcement learning in markov games. *Cognitive Systems Research*, 2(1):55–66.

[Liu et al., 2007] Liu, W., Winfield, A. F., Sa, J., Chen, J., and Dou, L. (2007). Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive behavior*, 15(3):289–305.

[Luke et al., 1998a] Luke, S. et al. (1998a). Genetic programming produced competitive soccer softbot teams for robocup97. *Genetic Programming*, 1998:214–222.

[Luke et al., 1998b] Luke, S., Hohn, C., Farris, J., Jackson, G., and Hendler, J. (1998b). Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: Robot soccer world cup I*, pages 398–411. Springer.

[Mallouk and Sen, 2009] Mallouk, T. E. and Sen, A. (2009). Powering nanorobots. *Scientific American*, 300(5):72–77.

[Matarić, 1997] Matarić, M. J. (1997). Reinforcement learning in the multi-robot domain. In *Robot colonies*, pages 73–83. Springer.

[Matarić, 2001] Matarić, M. J. (2001). Learning in behavior-based multi-robot systems: Policies, models, and other agents. *Cognitive Systems Research*, 2(1):81–93.

[Mataric et al., 1995] Mataric, M. J., Nilsson, M., and Simsarin, K. T. (1995). Cooperative multi-robot box-pushing. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 556–561. IEEE.

[Milutinovic and Lima, 2006] Milutinovic, D. and Lima, P. (2006). Modeling and optimal centralized control of a large-size robotic population. *IEEE Transactions on Robotics*, 22(6):1280–1285.

[Mondada et al., 2004] Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I. W., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L. M., and Dorigo, M. (2004). Swarm-bot: A new distributed robotic concept. *Autonomous robots*, 17(2-3):193–221.

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[Nehmzow, 2002] Nehmzow, U. (2002). Physically embedded genetic algorithm learning in multi-robot scenarios: The pega algorithm.

[Nelson et al., 2009] Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345 – 370.

[Nelson et al., 2004] Nelson, A. L., Grant, E., and Henderson, T. C. (2004). Evolution of neural controllers for competitive game playing with teams of mobile robots. *Robotics and Autonomous Systems*, 46(3):135–150.

[Nitschke et al., 2010] Nitschke, G. S., Schut, M. C., and Eiben, A. (2010). Collective neuro-evolution for evolving specialized sensor resolutions in a multi-rover task. *Evolutionary Intelligence*, 3(1):13–29.

[Nitschke et al., 2012] Nitschke, G. S., Schut, M. C., and Eiben, A. (2012). Evolving behavioral specialization in robot teams to solve a collective construction task. *Swarm and Evolutionary Computation*, 2:25–38.

[Nolfi and Floreano, 2000] Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press.

[O'Dowd et al., 2011] O'Dowd, P., Winfield, A. F., and Studley, M. (2011). The distributed co-evolution of an embodied simulator and controller for swarm robot behaviours.

[Olson, 2011] Olson, E. (2011). Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407.

[Panait and Luke, 2005] Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434.

[Parker, 1998] Parker, L. E. (1998). Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE transactions on robotics and automation*, 14(2):220–240.

[Parker, 2008] Parker, L. E. (2008). Multiple mobile robot systems. In *Springer Handbook of Robotics*, pages 921–941. Springer.

[Perez et al., 2008] Perez, A. L. F., Bittencourt, G., and Roisenberg, M. (2008). Embodied evolution with a new genetic programming variation algorithm.

In *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)*, pages 118–123. IEEE.

[Potter and De Jong, 2000] Potter, M. A. and De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, 8(1):1–29.

[Potter et al., 2001] Potter, M. A., Meeden, L. A., and Schultz, A. C. (2001). Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *International joint conference on artificial intelligence*, volume 17, pages 1337–1343. Citeseer.

[Prieto et al., 2010] Prieto, A., Becerra, J., Bellas, F., and Duro, R. J. (2010). Open-ended evolution as a means to self-organize heterogeneous multi-robot systems in real time. *Robotics and Autonomous Systems*, 58(12):1282–1291.

[Prieto et al., 2011] Prieto, A., Bellas, F., Caamaño, P., and Duro, R. (2011). Solving a heterogeneous fleet vehicle routing problem with time windows through the asynchronous situated coevolution algorithm. *Advances in artificial life. Darwin meets von Neumann*, pages 200–207.

[Prieto et al., 2016] Prieto, A., Bellas, F., Trueba, P., and Duro, R. (2016). Real-time optimization of dynamic problems through distributed embodied evolution. *Integrated Computer-Aided Engineering*, 23(3):237–253.

[Pugh and Martinoli, 2008] Pugh, J. and Martinoli, A. (2008). Distributed adaptation in multi-robot search using particle swarm optimization. *From Animals to Animats 10*, pages 393–402.

[Quinn, 2001] Quinn, M. (2001). A comparison of approaches to the evolution of homogeneous multi-robot teams. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 128–135. IEEE.

[Quinn et al., 2003] Quinn, M., Smith, L., Mayley, G., and Husbands, P. (2003). Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 361(1811):2321–2343.

[Russell and Norvig, 2010] Russell, S. J. and Norvig, P. (2010). Artificial intelligence (a modern approach).

[Schneider-Fontan and Mataric, 1998] Schneider-Fontan, M. and Mataric, M. J. (1998). Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation*, 14(5):815–822.

[Schwarzer et al., 2011] Schwarzer, C., Schlachter, F., and Michiels, N. K. (2011). Online evolution in dynamic environments using neural networks in autonomous robots. *International Journal On Advances in Intelligent Systems*, 4(3):288–298.

[Silva et al., 2015] Silva, F., Urbano, P., Correia, L., and Christensen, A. L. (2015). odneat: An algorithm for decentralised online evolution of robotic controllers. *Evolutionary Computation*, 23(3):421–449.

[Silva et al., 2012] Silva, F., Urbano, P., Oliveira, S., and Christensen, A. L. (2012). odneat: An algorithm for distributed online, onboard evolution of robot behaviours. *Artificial Life*, 13:251–258.

[Simmons et al., 2001] Simmons, R., Singh, S., Hershberger, D., Ramos, J., and Smith, T. (2001). First results in the coordination of heterogeneous robots for large-scale assembly. *Experimental Robotics VII*, pages 323–332.

[Simoes and Dimond, 1999] Simoes, E. and Dimond, K. R. (1999). An evolutionary controller for autonomous multi-robot systems. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 6, pages 596–601. IEEE.

[Soysal and Sahin, 2005] Soysal, O. and Sahin, E. (2005). Probabilistic aggregation strategies in swarm robotic systems. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 325–332. IEEE.

[Spears et al., 2006] Spears, D., Kerr, W., and Spears, W. (2006). Physics-based robot swarms for coverage problems. *International Journal on Intelligent Control and Systems*, 11(3):11–23.

[Spears et al., 2004] Spears, W. M., Spears, D. F., Hamann, J. C., and Heil, R. (2004). Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2-3):137–162.

[Stanley et al., 2005] Stanley, K. O., Bryant, B. D., and Miikkulainen, R. (2005). Real-time neuroevolution in the nero video game. *IEEE transactions on evolutionary computation*, 9(6):653–668.

[Stanley and Miikkulainen, 2002] Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

[Stone et al., 2005] Stone, P., Sutton, R. S., and Kuhlmann, G. (2005). Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior*, 13(3):165–188.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

[Takaya and Arita, 2003] Takaya, Y. U. and Arita, T. (2003). Situated and embodied evolution in collective evolutionary robotics. In *In Proc. of the 8th International Symposium on Artificial Life and Robotics*. Citeseer.

[Tanner and Kumar, 2005] Tanner, H. G. and Kumar, A. (2005). Towards decentralization of multi-robot navigation functions. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4132–4137.

[Trianni, 2008] Trianni, V. (2008). *Evolutionary swarm robotics: evolving self-organising behaviours in groups of autonomous robots*, volume 108. Springer.

[Trianni and Nolfi, 2009] Trianni, V. and Nolfi, S. (2009). Evolving collective control, cooperation and distributed cognition. *Handbook of Collective Robotics*, pages 127–166.

[Trianni and Nolfi, 2011] Trianni, V. and Nolfi, S. (2011). Engineering the evolution of self-organizing behaviors in swarm robotics: A case study. *Artificial Life*, 17(3):183–202.

[Trueba et al., 2013] Trueba, P., Prieto, A., Bellas, F., Caamaño, P., and Duro, R. J. (2013). Specialization analysis of embodied evolution for robotic collective tasks. *Robotics and Autonomous Systems*, 61(7):682–693.

[Trueba et al., 2011] Trueba, P., Prieto, A., Caamaño, P., Bellas, F., and Duro, R. J. (2011). Task-driven species in evolutionary robotic teams. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 138–147. Springer.

[Tuci et al., 2008] Tuci, E., Ampatzis, C., Vicentini, F., and Dorigo, M. (2008). Evolving homogeneous neurocontrollers for a group of heterogeneous robots:

Coordinated motion, cooperation, and acoustic communication. *Artificial Life*, 14(2):157–178.

[Tuci et al., 2006] Tuci, E., Groß, R., Trianni, V., Mondada, F., Bonani, M., and Dorigo, M. (2006). Cooperation through self-assembly in multi-robot systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2):115–150.

[Tuci and Trianni, 2014] Tuci, E. and Trianni, V. (2014). On the evolution of homogeneous two-robot teams: clonal versus aclonal approaches. *Neural Computing and Applications*, 25(5):1063–1076.

[Uchibe et al., 1998] Uchibe, E., Nakamura, M., and Asada, M. (1998). Co-evolution for cooperative behavior acquisition in a multiple mobile robot environment. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 1, pages 425–430. IEEE.

[Waibel et al., 2009] Waibel, M., Keller, L., and Floreano, D. (2009). Genetic team composition and level of selection in the evolution of cooperation. *IEEE Transactions on Evolutionary Computation*, 13(3):648–660.

[Walker et al., 2003] Walker, J., Garrett, S., and Wilson, M. (2003). Evolving controllers for real robots: A survey of the literature. *Adaptive Behavior*, 11(3):179–203.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[Watson et al., 2002] Watson, R., Ficici, S., and Pollack, J. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18. cited By 141.

[Watson et al., 1999] Watson, R. A., Ficici, S., and Pollack, J. B. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1. IEEE.

[Wendel et al., 2011] Wendel, A., Irschara, A., and Bischof, H. (2011). Natural landmark-based monocular localization for mavs. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5792–5799. IEEE.

[Werger and Matarić, 2000] Werger, B. B. and Matarić, M. J. (2000). Broadcast of local eligibility for multi-target observation. In *Distributed autonomous robotic systems 4*, pages 347–356. Springer.

[Wischmann et al., 2007] Wischmann, S., Stamm, K., and Wörgötter, F. (2007). Embodied evolution and learning: The neglected timing of maturation. In *European Conference on Artificial Life*, pages 284–293. Springer.

[Wolpert and Tumer, 2001] Wolpert, D. H. and Tumer, K. (2001). Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(02n03):265–279.

[Yang et al., 2008a] Yang, Z., Tang, K., and Yao, X. (2008a). Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 178(15):2985–2999.

[Yang et al., 2008b] Yang, Z., Tang, K., and Yao, X. (2008b). Multilevel cooperative coevolution for large scale optimization. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1663–1670. IEEE.

[Yong and Miikkulainen, 2001] Yong, C. H. and Miikkulainen, R. (2001). Cooperative coevolution of multi-agent systems. *University of Texas at Austin, Austin, TX*.

[Zlot and Stentz, 2006] Zlot, R. and Stentz, A. (2006). Market-based multirobot coordination for complex tasks. *The International Journal of Robotics Research*, 25(1):73–101.

Índice de figuras

Índice de tablas